ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО/МЭК 19784-1— 2007

Автоматическая идентификация

ИДЕНТИФИКАЦИЯ БИОМЕТРИЧЕСКАЯ

Биометрический программный интерфейс

Часть 1

Спецификация биометрического программного интерфейса

ISO/IEC 19784-1:2006
BioAPI — Biometric Application Programming Interface — Part 1: BioAPI Specification

(IDT)

Издание официальное

БЗ 5—2007/145

Москва
Стандартинформ
2009

gost.e-ecolog.ru

Предисловие

Цели и принципы стандартизации в Российской Федерации установлены Федеральным законом от 27 декабря 2002 г. № 184-ФЗ «О техническом регулировании», а правила применения национальных стандартов Российской Федерации — ГОСТ Р 1.0—2004 «Стандартизация в Российской Федерации. Основные положения»

Сведения о стандарте

1 ПОДГОТОВЛЕН Научно-исследовательским и испытательным центром биометрической техники Московского государственного технического университета имени Н. Э. Баумана (НИИЦ БТ МГТУ им. Н. Э. Баумана) на основе собственного аутентичного перевода стандарта, указанного в пункте 4 при консультационной поддержке Ассоциации автоматической идентификации «ЮНИСКАН/ГС1 РУС»

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 355 «Автоматическая идентификация»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 25 декабря 2007 г. № 402-ст

4 Настоящий стандарт идентичен международному стандарту ИСО/МЭК 19784-1:2006 «BioAPI. Биометрический программный интерфейс. Часть 1. Спецификация биометрического программного интерфейса» (ISO/IEC 19784-1:2006 «BioAPI — Biometric Application Programming Interface — Part 1: BioAPI Specification»), за исключением приложения G. Наименование настоящего стандарта изменено относительно наименования указанного международного стандарта для приведения в соответствие с ГОСТ Р 1.5 — 2004 (подраздел 3.5) и учета его принадлежности к группе стандартов «Автоматическая идентификация».

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты, сведения о которых приведены в дополнительном приложении G

5 ВВЕДЕН ВПЕРВЫЕ

Информация об изменениях к настоящему стандарту публикуется в ежегодно издаваемом информационном указателе «Национальные стандарты», а текст изменений и поправок — в ежемесячно издаваемых информационных указателях «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ежемесячно издаваемом информационном указателе «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет

© Стандартинформ, 2009

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения . . . 1
2 Соответствие . . . 2
3 Нормативные ссылки . . . 2
4 Термины и определения . . . 2
5 Обозначения и сокращения . . . 5
6 Архитектура БиоАПИ . . . 6
6.1 Архитектурная модель ПИП/ИПУ БиоАПИ . . . 6
6.2 Архитектурная модель ПБУ БиоАПИ . . . 7
6.3 Реестр компонентов . . . 8
6.4 Установка и удаление ПБУ и ПБФ . . . 9
6.5 Загрузка ПБУ и присоединение модуля БиоАПИ . . . 9
6.6 Управление модулями БиоАПИ . . . 10
6.7 Структура и обработка ЗБИ . . . 10
6.7.1 Структура ЗБИ . . . 10
6.7.2 Обработка данных ЗБИ . . . 11
7 Типы и макросы БиоАПИ . . . 12
7.1 Макрос БиоАПИ . . . 12
7.2 Тип BioAPI_BFP_LIST_ELEMENT . . . 12
7.3 Тип BioAPI_BFP_SCHEMA . . . 12
7.4 Тип BioAPI_BIR . . . 13
7.5 Тип BioAPI_BIR_ARRAY_POPULATION . . . 13
7.6 Тип BioAPI_BIR_BIOMETRIC_DATA_FORMAT . . . 13
7.7 Тип BioAPI_BIR_BIOMETRIC_PRODUCT_ID . . . 13
7.8 Тип BioAPI_BIR_BIOMETRIC_TYPE . . . 14
7.9 Тип BioAPI_BIR_DATA_TYPE . . . 14
7.10 Тип BioAPI_BIR_HANDLE . . . 15
7.11 Тип BioAPI_BIR_HEADER . . . 15
7.12 Тип BioAPI_BIR_PURPOSE . . . 16
7.13 Тип BioAPI_BIR_SECURITY_BLOCK_FORMAT . . . 17
7.14 Тип BioAPI_BIR_SUBTYPE . . . 17
7.15 Тип BioAPI_BOOL . . . 18
7.16 Тип BioAPI_BSP_SCHEMA . . . 18
7.17 Тип BioAPI_CANDIDATE . . . 19
7.18 Тип BioAPI_CATEGORY . . . 19
7.19 Тип BioAPI_DATA . . . 19
7.20 Тип BioAPI_DATE . . . 20
7.21 Тип BioAPI_DB_ACCESS_TYPE . . . 20
7.22 Тип BioAPI_DB_MARKER_HANDLE . . . 20
7.23 Тип BioAPI_DB_HANDLE . . . 20
7.24 Тип BioAPI_DBBIR_ID . . . 20
7.25 Тип BioAPI_DTG . . . 21
7.26 Тип BioAPI_EVENT . . . 21
7.27 Тип BioAPI_EVENT_MASK . . . 21
7.28 Тип BioAPI_EventHandler . . . 21
7.29 Тип BioAPI_FMR . . . 22
7.30 Тип BioAPI_FRAMEWORK_SCHEMA . . . 22
7.31 Тип BioAPI_GUI_BITMAP . . . 23
7.32 Тип BioAPI_GUI_MESSAGE . . . 23
7.33 Тип BioAPI_GUI_PROGRESS . . . 23
7.34 Тип BioAPI_GUI_RESPONSE . . . 24
7.35 Тип BioAPI_GUI_STATE . . . 24
7.36 Тип BioAPI_GUI_STATE_CALLBACK . . . 24
7.37 Тип BioAPI_GUI_STREAMING_CALLBACK . . . 25

7.38 Тип BioAPI_HANDLE . . . 25
7.39 Тип BioAPI_IDENTIFY_POPULATION . . . 25
7.40 Тип BioAPI_IDENTIFY_POPULATION_TYPE . . . 25
7.41 Тип BioAPI_INDICATOR_STATUS . . . 25
7.42 Тип BioAPI_INPUT_BIR . . . 25
7.43 Тип BioAPI_INPUT_BIR_FORM . . . 26
7.44 Тип BioAPI_INSTALL_ACTION . . . 26
7.45 Тип BioAPI_INSTALL_ERROR . . . 26
7.46 Тип BioAPI_OPERATIONS_MASK . . . 26
7.47 Тип BioAPI_OPTIONS_MASK . . . 27
7.48 Тип BioAPI_POWER_MODE . . . 28
7.49 Тип BioAPI_QUALITY . . . 28
7.50 Тип BioAPI_RETURN . . . 29
7.51 Тип BioAPI_STRING . . . 29
7.52 Тип BioAPI_TIME . . . 29
7.53 Тип BioAPI_UNIT_ID . . . 29
7.54 Тип BioAPI_UNIT_LIST_ELEMENT . . . 29
7.55 Тип BioAPI_UNIT_SCHEMA . . . 30
7.56 Тип BioAPI_UUID . . . 31
7.57 Тип BioAPI_VERSION . . . 31
8 Функции БиоАПИ . . . 31
8.1 Функции управления компонентом . . . 31
8.1.1 Функция BioAPI_Init . . . 31
8.1.2 Функция BioAPI_Terminate . . . 32
8.1.3 Функция BioAPI_GetFrameworkInfo . . . 32
8.1.4 Функция BioAPI_EnumBSPs . . . 32
8.1.5 Функция BioAPI_BSPLoad . . . 33
8.1.6 Функция BioAPI_BSPUnload . . . 34
8.1.7 Функция BioAPI_BSPAttach . . . 35
8.1.8 Функция BioAPI_BSPDetach . . . 36
8.1.9 Функция BioAPI_QueryUnits . . . 36
8.1.10 Функция BioAPI_EnumBFPs . . . 37
8.1.11 Функция BioAPI_QueryBFPs . . . 38
8.1.12 Функция BioAPI_ControlUnit . . . 38
8.2 Операции над дескриптором данных . . . 39
8.2.1 Операция BioAPI_FreeBIRHandle . . . 39
8.2.2 Операция BioAPI_GetBIRFromHandle . . . 40
8.2.3 Операция BioAPI_GetHeaderFromHandle . . . 40
8.3 Обратные вызовы и работа с событиями . . . 40
8.3.1 Функция BioAPI_EnableEvents . . . 40
8.3.2 Функция BioAPI_SetGUICallbacks . . . 41
8.4 Биометрические функции . . . 41
8.4.1 Функция BioAPI_Capture . . . 41
8.4.2 Функция BioAPI_CreateTemplate . . . 42
8.4.3 Функция BioAPI_Process . . . 44
8.4.4 Функция BioAPI_ProcessWithAuxBIR . . . 44
8.4.5 Функция BioAPI_VerifyMatch . . . 45
8.4.6 Функция BioAPI_IdentifyMatch . . . 46
8.4.7 Функция BioAPI_Enroll . . . 48
8.4.8 Функция BioAPI_Verify . . . 50
8.4.9 Функция BioAPI_Identify . . . 52
8.4.10 Функция BioAPI_Import . . . 54
8.4.11 Функция BioAPI_PresetIdentifyPopulation . . . 55
8.5 Операции над базой данных . . . 56
8.5.1 Функция BioAPI_DbOpen . . . 56

8.5.2 Функция BioAPI_DbClose . . . 57
8.5.3 Функция BioAPI_DbCreate . . . 57
8.5.4 Функция BioAPI_DbDelete . . . 58
8.5.5 Функция BioAPI_DbSetMarker . . . 58
8.5.6 Функция BioAPI_DbFreeMarker . . . 59
8.5.7 Функция BioAPI_DbStoreBIR . . . 59
8.5.8 Функция BioAPI_DbGetBIR . . . 59
8.5.9 Функция BioAPI_DbGetNextBIR . . . 60
8.5.10 Функция BioAPI_DbDeleteBIR . . . 60
8.6 Операции с модулями БиоАПИ . . . 61
8.6.1 Функция BioAPI_SetPowerMode . . . 61
8.6.2 Функция BioAPI_SetIndicatorStatus . . . 61
8.6.3 Функция BioAPI_GetIndicatorStatus . . . 62
8.6.4 Функция BioAPI_CalibrateSensor . . . 62
8.7 Служебные операции . . . 63
8.7.1 Функция BioAPI_Cancel . . . 63
8.7.2 Функция BioAPI_Free . . . 63
9 Интерфейс поставщика услуги БиоАПИ . . . 64
9.1 Общие положения . . . 64
9.2 Определение типов (для ПБУ) . . . 64
9.2.1 BioSPI_EventHandler . . . 64
9.2.2 BioSPI_BFP_ENUMERATION_HANDLER . . . 64
9.2.3 BioSPI_MEMORY_FREE_HANDLER . . . 65
9.3 Биометрические операции поставщика услуги . . . 66
9.3.1 Операции ИПУ управления компонентом . . . 66
9.3.2 Операции над дескриптором данных ИПУ . . . 68
9.3.3 Обратные вызовы и работа с событиями ИПУ . . . 69
9.3.4 Биометрические операции ИПУ . . . 69
9.3.5 Операции над базой данных ИПУ . . . 71
9.3.6 Операции с модулями БиоАПИ ИПУ . . . 72
9.3.7 Служебные операции ИПУ . . . 73
10 Интерфейс реестра компонентов . . . 73
10.1 Схема реестра компонентов БиоАПИ . . . 73
10.1.1 Схема инфраструктуры . . . 74
10.1.2 Схема ПБУ . . . 74
10.1.3 Схема ПБФ . . . 76
10.2 Функции реестра компонентов . . . 76
10.2.1 BioAPI_Util_InstallBsp . . . 76
10.2.2 BioAPI_Util_InstallBFP . . . 77
11 Обработка ошибок БиоАПИ . . . 77
11.1 Значения ошибок и схемы кодов ошибок . . . 78
11.2 Перечень кодов ошибок и значений ошибок . . . 78
11.2.1 Константы значения ошибки БиоАПИ . . . 78
11.2.2 Определяемые реализацией коды ошибок . . . 78
11.2.3 Общие коды ошибки . . . 78
11.2.4 Коды ошибок управления компонентом . . . 79
11.2.5 Значения ошибок базы данных . . . 79
11.2.6 Значения ошибок положения . . . 80
11.2.7 Коды ошибок качества . . . 81
Приложение А (обязательное) Соответствие . . . 82
Приложение В (обязательное) Формат постоянного клиента спецификации ЕСФОБД: формат постоянного клиента БиоАПИ . . . 92
Приложение С (справочное) Краткий обзор стандарта . . . 97
Приложение D (справочное) Примеры последовательности вызовов и типового кода . . . 105
Приложение Е (справочное) Сведения о соответствии ссылочных международных стандартов национальным стандартам . . . 117
Библиография . . . 118

Введение

Настоящий стандарт представляет собой спецификацию архитектуры (далее — модель) программного интерфейса для биометрических приложений (БиоАПИ) и определяет высокоуровневую обобщенную модель биометрического распознавания, пригодную для использования в любой биометрической технологии. Настоящий стандарт не обеспечивает поддержку мультимодальной биометрии в явном виде.

Описываемая модель позволяет использовать компоненты биометрической системы разных изготовителей и обеспечивать их взаимодействие посредством установленных программных интерфейсов приложений (ПИП).

Основой модели является инфраструктура БПИ (BioAPI Framework), поддерживающая вызовы одного или нескольких биометрических программных приложений (далее — приложений), которые могут быть предоставлены различными изготовителями и потенциально могут выполняться одновременно с помощью ПИП БПИ. Инфраструктура БПИ обеспечивает поддержку вызова с помощью интерфейса поставщика услуги (ИПУ) одного или нескольких поставщиков биометрической услуги (ПБУ), которые могут быть предоставлены различными изготовителями и, возможно, будут выполняться одновременно, а также могут быть динамически загружены и вызваны в соответствии с требованиями биометрического приложения.

На самом нижнем уровне модели находятся аппаратные или программные средства, выполняющие биометрические функции, например, такие как получение данных, сравнение или архивирование. Такие составляющие биометрической системы называются модулями БиоПИП и могут являться частью ПБУ или поставляться как компонент поставщика функции БиоПИП (ПФБ).

Между ПБУ различных изготовителей, предоставляющих для записи информации от модулей БиоАПИ, к которым они имеют доступ, структуры данных, соответствующие другим международным стандартам, особенно ISO/IEC 19794, могут быть установлены взаимодействия через инфраструктуру БиоАПИ.

Таким образом, ПБУ может обеспечивать выполнение биометрических услуг с помощью:

a) модулей БиоАПИ, которые являются частью или непосредственно управляются ПБУ, или

b) вызова через интерфейс поставщика функции БиоПИП (ИПФ) одного или нескольких компонентов поставщика биометрической функции (ПБФ) разных изготовителей, которые управляют модулями БиоАПИ, являющимися частью ПБФ.

Примечание — Модуль БиоАПИ может состоять только из программного обеспечения или из комбинации программного и аппаратного обеспечения (например, биометрический сканер, архив или алгоритм).

Допускается наличие одного или более модулей БиоПИП определенного типа, который поддерживается ПБУ (или ПБФ). Данные модули могут быть динамически добавлены и удалены из системы. Добавление и удаление вызывают события, о которых может быть сообщено приложению (через ПБУ и инфраструктуру БиоПИП).

Настоящий стандарт устанавливает основные биометрические функции регистрации, верификации и идентификации (приложение С) и включает в себя интерфейс базы данных, что позволяет приложению управлять хранением биометрических записей с помощью модуля архива БиоАПИ, управляемого ПБУ или ПБФ. Это обеспечивает оптимальную производительность процессов архивирования и сравнения биометрического образца с множеством контрольных шаблонов (например, при выполнении функции биометрической идентификации по большой выборке).

Интерфейс предоставляет примитивы*, которые позволяют приложению управлять получением биометрических образцов от биометрических сканеров с помощью доступа к соответствующему модулю БиоПИП, использованием этих биометрических образцов для регистрации (хранения в базе записей биометрической информации (ЗБИ), управляемой приложением или ПБУ), и последующей верификацией и идентификацией.

Настоящий стандарт также устанавливает содержание реестра компонентов (информацию о биометрических компонентах, установленных в биометрической системе). Он также обеспечивает интерфейс реестра компонентов для управления и проверки данного реестра.

В настоящем стандарте использован язык программирования Си (по ИСО/МЭК 9899:1999 «Языки программирования. Си») для определения структур данных и вызовов функций, которые формируют интерфейсы БиоАПИ.

---

* *В настоящем стандарте термин «примитивы» обозначает базовый элемент, используемый для построения программы.*

Раздел 6 устанавливает модель БиоАПИ, ее компоненты и интерфейсы, определенные для этих компонентов.

Раздел 7 устанавливает структуры данных, используемые в БиоАПИ.

Раздел 8 устанавливает вызовы функций, инициируемые приложением и поддерживаемые соответствующей инфраструктурой БиоАПИ, которые обрабатываются либо непосредственно инфраструктурой БиоАПИ (например, получение списков установленных компонентов БиоАПИ), либо переадресовываются функциям, предоставляемым ПБУ.

Раздел 9 устанавливает вызовы функций, поддерживаемые соответствующим ПБУ и осуществляемые инфраструктурой БиоАПИ в ответ на вызов биометрического приложения.

Раздел 10 устанавливает форму реестра биометрических компонентов и интерфейс реестра компонентов.

Раздел 11 устанавливает обработку событий и возвращаемые ошибки.

В приложении А приведены подробные требования и таблицы определения соответствия спецификации БиоПИП, которые могут быть использованы разработчиком компонентов приложения БиоПИП, инфраструктуры БиоПИП или ПБУ для определения поддерживаемых функций и биометрических форматов записи.

Примечание — Методы тестирования на соответствие БиоАПИ требованиям настоящего стандарта приведены в ISO/IEC 24709.

В приложении В приведено описание ЗБИ БиоАПИ, соответствующее формату ведущей организации, определенному в ИСО/МЭК 19785-1 «Информационные технологии. Единая структура формата обмена биометрическими данными. Часть 1. Спецификация элементов данных». В приложении также приведено описание биометрической записи, установленное в вышеуказанном стандарте, и побитовое представление записи для ее хранения и передачи.

В приложении С приведено краткое описание основных положений настоящего стандарта.

В приложении D приведен пример текста программы для установления последовательности вызовов функций.

Отдельные положения стандарта ИСО/МЭК 19784-1 могут быть предметом других патентных прав, помимо указанных в нем.

Сноски в тексте стандарта приведены для пояснения и выделены курсивом.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Автоматическая идентификация

ИДЕНТИФИКАЦИЯ БИОМЕТРИЧЕСКАЯ

Биометрический программный интерфейс

Часть 1

Спецификация биометрического программного интерфейса

Automatic identification. Biometrics. BioAPI. Biometric Application Programming Interface. Part 1. BioAPI Specification

Дата введения — 2009—01—01

1 Область применения

Настоящий стандарт устанавливает программный интерфейс приложения (ПИП) и интерфейс поставщика услуги (ИПУ) в качестве стандартных интерфейсов биометрической системы, позволяющих создавать биометрическую систему, используя компоненты, изготовленные различными изготовителями. Соответствие ПИП и ИПУ требованиям настоящего стандарта и других стандартов ИСО/МЭК обеспечивает взаимодействие между этими компонентами.

Положения настоящего стандарта могут применяться для разных биометрических технологий и систем различного назначения: от персональных устройств идентификации и систем обеспечения сетевой безопасности до масштабных комплексных систем идентификации.

Настоящий стандарт устанавливает модель, в которой инфраструктура БиоАПИ, наряду с поддержкой нескольких биометрических приложений (возможно предоставленных различными изготовителями), может:

- использовать несколько динамически устанавливаемых, загружаемых или выгружаемых компонентов ПБУ (возможно изготовленных на различных предприятиях) и через них — модули БиоПИП,
- использовать один из альтернативных наборов компонентов ПБФ (возможно изготовленных на различных предприятиях);
- использовать непосредственно модули БиоАПИ.

Примечание — Если модули БиоАПИ и ПБУ предоставлены разными изготовителями, может потребоваться использование ИПФ, требования к которому установлены в следующих частях комплекса стандартов ИСО/МЭК 19784.

Настоящий стандарт не распространяется на биометрические системы, разработанные одним изготовителем, в частности, когда для них не предусмотрена возможность добавления или замены программных и/или аппаратных средств или изменения целей использования.

Настоящий стандарт не устанавливает требований безопасности для биометрических приложений и ПБУ.

Примечание — Руководство по аспектам безопасности биометрических систем приведено в ISO/IEC 19092 [3].

Настоящий стандарт не распространяется на эксплуатационные характеристики биометрических систем (особенно идентификационных систем) и на зависимость эксплуатационных характеристик системы от функциональной совместимости компонентов.

Настоящий стандарт определяет версию спецификации БиоАПИ с номером последней редакции 2 и номером поправки (или номер изменения данной редакции) 0, то есть версию 2.0.

Примечание — Настоящий стандарат не распространяется на более ранние версии спецификации БиоАПИ.

Издание официальное

2 Соответствие

2.1 Компоненты БиоПИП считают соответствующими требованиям настоящего стандарта, если они соответствуют требованиям приложения А.

2.2 Для определения описываемых интерфейсов в настоящем стандарте использован язык программирования Си по ИСО/МЭК 9899. Компонент БиоАПИ, предоставляющий или использующий интерфейс на других языках программирования, может соответствовать требованиям настоящего стандарта при условии, что взаимодействующий с ним компонент системы может использовать интерфейс с помощью подробной спецификации на языке программирования Си, представленной в настоящем стандарте (7.1).

3 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты, которые необходимо учитывать при использовании настоящего стандарта. В случае ссылок на стандарты, у которых указана дата утверждения, необходимо пользоваться только указанной редакцией. В случае, когда дата утверждения не приведена, следует пользоваться последней редакцией ссылочных стандартов, включая любые поправки и изменения к ним.

ИСО/МЭК 9834-8 Информационные технологии. Взаимосвязь открытых систем. Процедуры для работы регистрационных органов в системе OSI. Часть 8. Создание и регистрация универсальных уникальных идентификаторов (UUIDs) и их использование в качестве компонентов идентификаторов объектов ASN1.

ИСО/МЭК 9899:1999 Языки программирования. Си

ИСО/МЭК 10646-2003 Информационные технологии. Универсальный многоактетный комплекс закодированных знаков (UCS).

ИСО/МЭК 19785-1 Информационные технологии. Единая структура форматов обмена биометрическими данными. Часть 1. Спецификация элементов данных

ИСО/МЭК 19785-2 Информационные технологии. Единая структура форматов обмена биометрическими данными. Часть 2. Процедуры действий регистрационного органа в области биометрии

4 Термины и определения

В настоящем стандарте применены следующие термины и определения:

Примечание — Названия функций и элементов данных не включены в данный раздел, а их определения приведены в тексте настоящего стандарта.

4.1 **адаптация; адаптация шаблона** (adaptation; template adaptation): Использование ЗБИ, полученной с вновь зарегистрированного и верифицированного биометрического образца, для автоматического обновления или модернизации зарегистрированного ранее контрольного шаблона.

Примечание — Данную процедуру используют для минимизации эффектов устаревания шаблонов.

4.2 **присоединенная сессия** (attach session): Временная связь между приложением, отдельным ПБУ и набором модулей, напрямую или косвенно управляемых ПБУ.

4.3 **компонент БиоАПИ** (BioAPI component): Компонент архитектуры БиоАПИ с определенным интерфейсом, который может быть предоставлен отдельным изготовителем, используемый при испытании на соответствие.

Примечание — Компоненты БиоАПИ включают в себя приложения БиоАПИ, инфраструктуру БиоАПИ, ПБУ и ПБФ.

4.4 **поставщик функции БиоАПИ;** ПБФ (BioAPI function provider; BFP): Компонент, управляющий одним или более модулями БиоАПИ определенной категории.

Примечание 1 — Интерфейсы ПБФ стандартизованы в следующих частях комплекса стандартов ИСО/МЭК 19784.

Примечание 2 — ПБУ распределены на типы, соответствующие типам модулей БиоАПИ, которыми они управляют (см. 6.2.4).

4.5 **модуль БиоАПИ** (BioAPI unit): Абстракция аппаратного или программного уровня, напрямую управляемая ПБУ или ПБФ.

Примечание — Модули БиоАПИ категоризированы (6.2.2) и включают в себя модули сканеров, архива, алгоритмов сравнения и алгоритмов обработки.

4.6 **биометрический** (biometric): Имеющий отношение к биометрии.

4.7 **блок биометрических данных;** ББД (biometric data block; BDB): Блок данных, построенный в соответствии с определенным форматом и содержащий один или несколько биометрических образцов или биометрических шаблонов.

Примечание 1 — Настоящий стандарт не распространяется на форматы, допускающие, что размер ББД может не быть кратным восьми битам.

Примечание 2 — Требования к структуре формата ББД не предъявляются.

Примечание 3 — Каждая* часть комплекса ИСО/МЭК 19794 устанавливает требования к одному и более форматам ББД. Изготовитель может создать и зарегистрировать** свой формат.

Примечание 4 — Информация, содержащаяся в ББД, не имеет значения для приложения БиоАПИ, поэтому далее в тексте стандарта использован термин «неопределенный блок биометрических данных».

4.8 **запись биометрической информации;** ЗБИ (biometric information record; BIR): Структура данных, включающая в себя один или более ББД вместе с информацией, идентифицирующей форматы ББД, и дополнительной информацией, например о типе ББД (подписанный, зашифрованный).

Примечание — Настоящий стандарт устанавливает формат ЗБИ (см. 7.4), включающий в себя только один ББД. ИСО/МЭК 19785 устанавливает формат ЗБИ, содержащий несколько ББД в ЗБИ, однако в нем приведено аналогичное определение данного термина. Необходимо учитывать, что при использовании термина ЗБИ в тексте настоящего стандарта имеется в виду ЗБИ, соответствующая требованиям приложения В. При необходимости дополнительных пояснений в тексте стандарта может быть использован термин «БиоАПИ ЗБИ».

4.8.1 **контрольная ЗБИ** (reference BIR): ЗБИ, ББД которой содержит один или более биометрических шаблонов.

4.8.2 **образец ЗБИ** (sample BIR). ЗБИ, ББД которой содержит только биометрические образцы, не являющиеся шаблонами.

4.9 **биометрический образец** (biometric sample): Информация с биометрического сканера, полученная непосредственно или после обработки.

Примечание — См. также терминологические статьи 4.9.2, 4.9.3 и 4.9.4.

4.9.1 **биометрический шаблон** (biometric template): Биометрический образец или комбинация биометрических образцов, пригодных для хранения в качестве контрольных для дальнейшего сравнения.

4.9.2 **промежуточный биометрический образец** (intermediate biometric sample): Биометрический образец, полученный путем обработки исходного биометрического образца и предназначенный для дальнейшей обработки.

4.9.3 **обработанный биометрический образец** (processed biometric sample): Биометрический образец, предназначеный для сравнения.

4.9.4 **исходный биометрический образец** (raw biometric sample): Биометрический образец, полученный непосредственно с биометрического сканера.

Примечание — В настоящее время стандарты, устанавливающие требования к форматам необработанных биометрических образцов, не разработаны. Спецификации данных форматов зависят от характеристик биометрических сканеров и от их изготовителей. При разработке стандартов на биометрические сканеры должны быть установлены требования к форматам необработанных биометрических образцов.

4.9.5 **контрольный шаблон** (reference template): Сохраненный биометрический шаблон.

4.10 **биометрический сканер** (biometric sensor): Биометрическое аппаратное средство, используемое для получения исходных биометрических образцов.

Примечание — Также допускается использовать термин «биометрическое устройство».

4.11 **поставщик биометрической услуги;** ПБУ (biometric service provider; BSP): Компонент, осуществляющий для приложения определенные действия либо с помощью определенного интерфейса путем непосредственного управления одним или несколькими модулями БиоАПИ, либо через поставщиков функции БиоАПИ также с помощью определенного интерфейса.

** В оригинале ИСО/МЭК 19784-1 допущена ошибка. Первая часть этого стандарта называется «Принципы и структура» и не устанавливает требований к какому-либо формату ББД.*

*** Регистрация должна быть проведена органом регистрации в области биометрии в системе ЕСФОБД с присвоением соответствующих идентификаторов по ГОСТ Р ИСО/МЭК 19785-1 — 2007 и ГОСТ Р ИСО/МЭК 19785-2 — 2007.*

4.12 **биометрия** (biometrics): Автоматическое* распознавание индивидума, в основе которого лежат его поведенческие и биологические характеристики.

4.13 **обратный вызов** (callback): Механизм, в соответствии с которым компонент, предоставляющий ПИП, вызывает функцию из компонента, использующего ПИП, причем адрес данной функции предварительно передан в качестве входного параметра при вызове функции ПИП.

Примечание — Данный механизм позволяет компоненту БиоАПИ связаться с другим компонентом БиоАПИ, не используя вызовы функции ПИП, обычно в ответ на произошедшее событие или прерывание.

4.14 **реестр компонентов** (component registry): Информация, сохраняемая инфраструктурой БиоАПИ, о доступных в биометрической системе компонентах БиоАПИ.

4.15 **шифровать/шифрование** (encrypt/encryption): Обратимое преобразование данных с помощью криптографического алгоритма для создания зашифрованного** текста с целью защиты информации (обеспечения конфиденциальности).

Примечание 1 — Алгоритмы шифрования обеспечивают выполнение двух процессов: шифрование, которое преобразует исходный текст в зашифрованный, и дешифрование, которое преобразует зашифрованный текст в исходный.

Примечание 2 — Шифрование может использоваться как с целью обеспечения безопасности, так и с целью обеспечения конфиденциальности.

4.16 **регистрация** (enrollment): Процесс получения одного или нескольких биометрических образцов человека с последующим построением биометрического контрольного шаблона, используемого для верификации или идентификации личности.

Примечание — Контрольный шаблон обычно сохраняется биометрическим приложением и/или ПБУ, поддерживающим модуль архива БиоАПИ.

4.17 **вероятность ошибки ложного совпадения;** ОЛС (false match rate; FMR): Мера вероятности, с которой процесс биометрического сравнения неверно идентифицирует личность или не сможет отказать в доступе нарушителю.

Примечание 1 — В БиоАПИ ОЛС используется в качестве средства установления оценок соответствия и порогов (приложение С).

Примечание 2 — Раньше данное определение использовалось также для определения термина «вероятность ошибки ложного доступа», но использование термина «вероятность ошибки ложного совпадения» является предпочтительным. Для терминов «вероятность ошибки ложного отказа; ОЛО» и «вероятность ошибки ложного несовпадения; ОЛН» использование последнего является предпочтительным*.

4.18 **дескриптор** (handle): Параметр, возвращаемый функцией БиоАПИ (например, А), который может быть использован приложением БиоАПИ в следующем вызове функции для идентификации компонента БиоАПИ или элемента данных в рамках компонента А.

Примечание — Типами дескрипторов являются:

BIR_Handle — создается ПБУ для выбора или доступа к ЗБИ в рамках данного ПБУ;

BSP Attach Session Handle — используется для присоединенной сессии;

DB_Handle — создается ПБУ для выбора или доступа к управляемой данным ПБУ базе данных ЗБИ.

4.19 **идентифицировать/идентификация** (identify/identification): Процесс сравнения предложенного биометрического образца с контрольной выборкой (схема «один ко многим») с целью определения личности, которой соответствует определенный шаблон данной выборки, имеющий сходство с предложенным биометрическим образцом.

Примечание — Данный процесс часто называют «идентификационное сопоставление».

4.20 **сопоставлять/сопоставление** (match/matching): Процесс сравнения предложенного биометрического образца с одним из контрольных биометрических шаблонов (схема «один к одному») с целью оценки степени схожести.

Примечание 1 — Основой решения о допуске или об отказе обычно является превышение данной оценки степени схожести порогового значения.

Примечание 2 — Алгоритмы сопоставления и их влияние на вероятность ошибки ложного совпадения и вероятность ошибки ложного несовпадения в настоящее время не стандартизованы.

Примечание 3 — Также см. определения терминов «идентифицировать» (4.19) и «верифицировать» (4.28).

---

* *В том числе и автоматизированное.*

** *Термин «зашифрованный» может быть определен как защищенный, закрытый.*

4.21 **полезная информация** (payload): Данные, собранные во время регистрации и относящиеся к контрольному шаблону, которые могут быть выданы после удачной биометрической верификации.

Примечание — Примерами полезной информации являются имена пользователей, счета, пароли, криптографические ключи, цифровые подписи (приложение С).

4.22 **оценка схожести/оценивание схожести** (score/scoring): Значение, определяющее степень соответствия биометрического образца и контрольного биометрического шаблона.

4.23 **блок безопасности** (security block): Блок данных, содержащий информацию о целостности данных ЗБИ.

4.24 **независимое устройство** (self-contained device): Комбинированное устройство, которое включает в себя биометрический сканер, а также все или часть функций ПБУ.

Примечание — Независимое устройство может не только получать биометрические образцы, но также обрабатывать, сопоставлять и/или хранить их. Данные функции обычно реализованы на аппаратном уровне или на уровне встроенных программных средств.

4.25 **подпись/цифровая подпись** (signature/digital signature): Особые данные, добавленные к блоку данных, или криптографическое преобразование блока данных, позволяющее получателю блока данных гарантировать подлинность автора и целостность блока данных и защитить данные от фальсификации.

Примечание — Цифровые подписи могут также использоваться для определения подлинности автора, обеспечения целостности данных и невозможности отказаться от авторства.

4.26 **порог** (threshold): Значение, устанавливающее степень схожести или корелляции, превышение которого позволяет считать биометрический образец соответствующим контрольному биометрическому шаблону.

4.27 **универсальный уникальный идентификатор;** УУИД (universally unique identificator; UUID): 128-битовое значение, определенное по ИСО/МЭК 9834-8.

4.28 **верифицировать/верификация** (verify/verification): Процесс сравнения биометрического образца с контрольным биометрическим шаблоном (схема «один к одному») с целью определения соответствия предложенного биометрического образца контрольному шаблону.

Примечание — Данный процесс также часто называют «верификационное сопоставление».

5 Обозначения и сокращения

В настоящем стандарте приняты следующие обозначения и сокращения:

ПИП — программный интерфейс приложений (Application Programming Interface; API);

ББД — блок биометрических данных (Biometric Data Block; BDB);

ПБФ — поставщик функции БиоАПИ (BioAPI Function Provider; BFP);

ПБУ — поставщик биометрической услуги (Biometric Service Provider; BSP);

ЕСФОБД — единая структура формата обмена биометрическими данными (Common Biometric Exchange Formats Framework; CBEFF);

ОЛС — вероятность ошибки ложного совпадения (False Match Rate; FMR);

ИПФ — интерфейс поставщика функции (Function Provider Interface; FPI),

ГИП — графический интерфейс пользователя (Graphical User Interface; GUI);

ИД — идентичность/идентификация/идентификатор (Identity/Identification/Identifier; ID);

СНК — сопоставление на карте (Match on Card; MOC);

ИДП — идентификатор продукта (Product ID; PID);

БЗИ — блок защиты информации (Security Block; SB);

СБЗ — стандартный биометрический заголовок (Standard Biometric Header; SBH).

Примечание — Данное сокращение соответствует приведенному в ИСО/МЭК 19785-1;

ИПУ — интерфейс поставщика услуги (Service Provider Interface);

УУИД — универсальный уникальный идентификатор (Universally Unique Identifier; UUID).

6 Архитектура БиоАПИ

6.1 Архитектурная модель ПИП/ИПУ БиоАПИ

6.1.1 БиоАПИ включает в себя модель ПИП/ИПУ, изображенную на рисунке 1.



Рисунок 1 — Модель БиоАПИ ПИП/ПБУ

Примечание — На рисунке 1 структура модели ниже уровня ИПУ упрощена. Возможные варианты внутренней структуры и стандартизированный интерфейс ПБУ изображены на рисунке 2.

6.1.2 Система БиоАПИ состоит из компонентов БиоАПИ, имеющих стандартизированные интерфейсы. На рисунке 1 показаны взаимодействия между тремя видами компонентов БиоАПИ: инфраструктурой БиоАПИ, приложениями и монолитными ПБУ.

Примечание — Описание стандартизированных интерфейсов, позволяющее независимым компонентам БиоАПИ предоставлять доступ к функциям ПБУ, приведено в 6.2.

6.1.3 ПИП определяет интерфейс между инфраструктурой БиоАПИ и биометрическим приложением. Приложение предназначено для вызова функций, описанных в спецификации ПИП (раздел 8). Инфраструктура БиоАПИ поддерживает функции, приведенные в спецификации ПИП.

6.1.4 ИПУ определяет интерфейс между инфраструктурой БиоАПИ и ПБУ. Инфраструктура БиоАПИ вызывает функции, описанные в спецификации ИПУ (раздел 9). ПБУ поддерживает функции, приведенные в спецификации ИПУ.

6.1.5 Инфраструктура БиоАПИ предназначена для управления ПБУ и отображения вызовов функций ПИП функциям ИПУ, адресованных соответствующему ПБУ.

6.1.6 Приложение может получить доступ к функциональным возможностям ПБУ (через инфраструктуру БиоАПИ) только после того как ПБУ будет загружен и присоединен (8.1.5 и 8.1.7). Когда приложению больше не требуется использование ПБУ, оно его отсоединяет и выгружает (см. 8.1.6 и 8.1.8).

6.1.7 Приложение может одновременно загружать и присоединять более одного ПБУ. ПБУ может быть одновременно присоединен к более чем одному приложению.

Примечание — Взаимодействия между приложением и присоединенным ПБУ обычно не зависят от взаимодействий между этим приложением и другими ПБУ или между этим ПБУ и другими приложениями, за исключением случаев, когда возникает конфликт с физическим устройством, управляемым ПБУ.

6.1.8 Функции, определенные в настоящем стандарте, поддерживают наличие в биометрической системе:

a) отдельной инфраструктуры БиоАПИ со связанным с ней реестром компонентов;

b) динамического выполнения и завершения нескольких одновременно выполняющихся биометрических приложений, взаимодействующих с данной инфраструктурой БиоАПИ;

c) динамической установки, удаления (и связанными с ними загрузкой и выгрузкой) нескольких ПБУ, взаимодействующих с данной инфраструктурой БиоАПИ;

d) сообщения от ПБУ инфраструктуре БиоАПИ (и следовательно, выполняющемуся биометрическому приложению) о событиях, связанных с динамическим присоединением и отсоединением модулей БиоАПИ (см. 6.5), управляемых данным ПБУ.

Примечание — Ожидается, что ПИП будет связан с единственным приложением, управляющим одним ПБУ с не более чем одним модулем БиоАПИ каждой биометрической категории, доступ к которому обеспечивается с помощью данного ПБУ. Тем не менее поддержка доступа приложения к нескольким ПБУ, каждый из которых способен управлять несколькими модулями БиоАПИ (но только одним из каждой категории для данной присоединенной сессии), помогает облегчить такие применения, как контроль физического доступа, особенно при использовании сетевых устройств.

6.2 Архитектурная модель ПБУ БиоАПИ

6.2.1 Модули БиоАПИ представляют собой абстракцию биометрических устройств и являются основными структурными блоками, которые предоставляет ПБУ приложению. Они включают в себя и скрывают программные и аппаратные ресурсы различных типов, такие как устройство регистрации, архивирующее устройство и т. д.

6.2.2 Каждый модуль БиоАПИ моделирует или включает в себя один (или ни одного) элемент аппаратного обеспечения и любое необходимое программное обеспечение. В настоящее время определенны следующие категории модулей БиоАПИ:

- модуль сканера;
- модуль архива;
- модуль алгоритма сопоставления;
- модуль алгоритма обработки.

Примечание — Первые два модуля, как правило, но не обязательно, связаны со аппаратными средствами, а последние два модуля, как правило, но не обязательно, не имеют связанных с ними аппаратных средств.

6.2.3 Модуль БиоАПИ может управляться ПБУ внутренне (прямое управление модулем БиоАПИ) или связанным поставщиком функции БиоАПИ (ПБФ) (непрямое управление модулем БиоАПИ). Интерфейсы поставщика функции БиоАПИ (ИПФ) определены в следующих частях комплекса стандартов ИСО/МЭК 19784.

6.2.4 ПБФ может управлять несколькими модулями БиоАПИ данной категории (но не более чем одним в любой присоединенной сессии для данного приложения). ПБФ категоризированы в соответствии с категориями модулей БиоАПИ, которыми они могут управлять. В настоящее время определены следующие категории ПБФ:

- сканера;
- архива;
- алгоритма сопоставления;
- алгоритма обработки.

6.2.5 ИПФ БиоАПИ для каждой категории ПБФ определены в других частях комплекса стандартов ИСО/МЭК 19784. Все ИПФ поддерживают доступ к одному или нескольким (но не более чем к одному в любой присоединенной сессии для данного приложения) модулям БиоАПИ в адресуемом ПБФ.

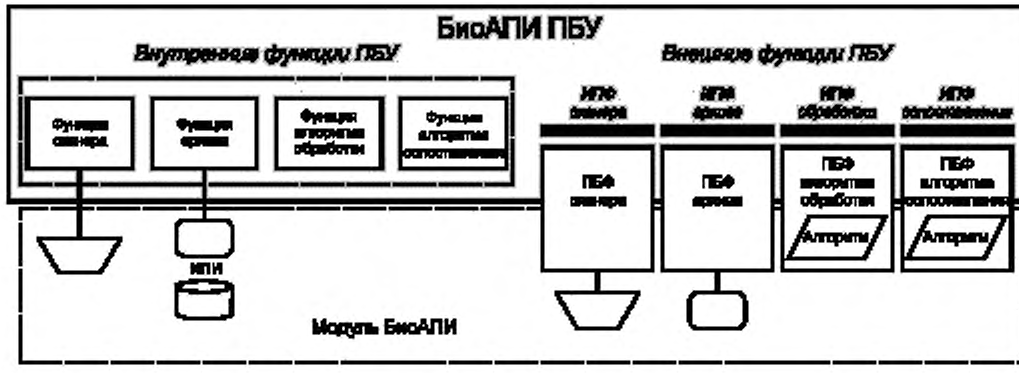


Рисунок 2 — Иллюстрация архитектуры ПБУ

6.2.6 ПБУ, поддерживающий несколько модулей БиоАПИ (управляемых непосредственно или с помощью ПБФ), может поддерживать интерфейс ИПУ, позволяющий приложению выбрать конкретный модуль БиоАПИ (один для каждой категории) для присоединенной сессии. Приложение может задать параметр BioAPI_DONT_CARE для модуля БиоАПИ конкретной категории. В этом случае выбор используемого модуля БиоАПИ осуществляет ПБУ.

6.2.7 Если разработчик ПБУ указывает конкретную категорию модуля БиоАПИ, то ПБУ должен иметь возможность управлять модулем БиоАПИ данной категории напрямую или взаимодействовать с ПБФ соответствующей категории с помощью соответствующего стандартизованного ИПФ.

6.2.8 Функции ПИП БиоАПИ и ИПУ (в некоторых случаях — функции ИПФ) (8.1.12 и 9.3.1.7) дают возможность приложению посылать модулю или запрашивать у модуля БиоАПИ управляющую информацию и информацию о статусе с использованием ПИП БиоАПИ и ИПУ (или ИПФ). Эти функции являются дополнительными к обычным биометрическим операциям. Параметры управляющих функций не стандартизованы. Если ПБУ или ПБФ (если он задействован) не поддерживает эту управляющую функцию, возвращается ошибка (6.6).

6.2.9 Если установлена присоединенная сессия ПБУ, выбирается не более чем один модуль БиоАПИ каждой категории (8.1.7).

Примечание — Для данной архитектуры требуется, чтобы интерфейсы ИПФ обеспечивали возможность ПБУ передавать ПБФ идентификацию (обеспечиваемую интерфейсом ПИП и ИПУ) того модуля БиоАПИ, который должен использоваться. Это подразумевает наличие конкретной камеры, сканера, устройства хранения и других устройств. Необходимые функции ИПФ для каждой категории ПБФ установлены в следующих частях комплекса стандартов ИСО/МЭК 19784.

6.3 Реестр компонентов

6.3.1 Реестр компонентов содержит информацию об установленных ПБУ и ПБФ.

6.3.2 В модели БиоАПИ предполагается наличие в биометрической системе единственного экземпляра инфраструктуры БиоАПИ и единственного связанного с ним реестра компонентов.

Примечание — В реальной информационной системе могут быть несколько реестров компонентов, поддерживаемых одним и тем же, но разделяемым, кодом инфраструктуры БиоАПИ либо разными кодами инфраструктуры БиоАПИ (возможно, старой и новой версиями инфраструктуры). Это обеспечивается наличием нескольких биометрических систем в одной реальной информационной системе.

6.3.3 БиоАПИ требует, чтобы не было взаимодействий или конфликтов между различными биометрическими системами, входящими в одну реальную компьютерную систему.

Примечание — Если реальная компьютерная система содержит несколько биометрических систем, возможное разделение кода и программно-алгоритмических средств, с помощью которых приложение связывается с той или иной биометрической системой, является вопросом исполнения.

6.3.4 Следующая информация может быть получена приложением с помощью функций инфраструктуры БиоАПИ, возвращающих информацию реестра компонентов (8.1.3, 8.1.4 и 8.1.10):

a) информация о самой инфраструктуре BioAPI;

b) подробные данные о всех установленных ПБУ;

c) подробные данные о всех установленных ПБФ.

Примечание — Информация об установленных ПБФ также может быть получена ПБУ с помощью механизма обратного вызова.

6.3.5 Следующая информация может быть получена приложением с помощью функций инфраструктуры БиоАПИ, передаваемых через ИПУ конкретному ПБУ (8.1.11 и 8.1.9):

a) подробные данные о всех установленных ПБФ, которые поддерживаются данным ПБУ;

b) подробные данные о всех модулях БиоАПИ, находящихся в подключенном состоянии, к которым возможен доступ с помощью данного ПБУ (напрямую или с помощью поддерживаемого ПБФ).

6.3.6 Информация, приведенная в 6.3.4 и 6.3.5, может быть получена приложением путем использования функций, которые могут быть вызваны в следующих условиях:

a) информация о самой инфраструктуре может быть получена в любое время после вызова функции ***BioAPI_Init*** (8.1.3);

b) подробные данные о всех установленных ПБУ могут быть получены в любое время после вызова функции ***BioAPI_Init*** (8.1.4);

c) подробные данные о всех установленных ПБФ могут быть получены в любое время после вызова функции ***BioAPI_Init*** (8.1.10);

d) подробные данные о всех установленных ПБФ, которые поддерживаются каждым ПБУ, могут быть получены в любое время после вызова функции ***BioAPI_Load*** (8.1.11) для этого ПБУ.

Примечание 1 — Приложением могут быть загружены одновременно несколько ПБУ.

Примечание 2 — Предполагается, что при вызове функции ***BSP_Load*** ПБУ использует ИПФ для загрузки всех ПБФ, которые он может использовать;

e) подробные данные о всех модулях БиоАПИ, находящихся в подключенном состоянии, к которым возможен доступ с помощью данного ПБУ (напрямую или с помощью поддерживаемого ПБФ), могут быть получены (по ссылке на УУИД ПБУ) в любое время после вызова функции ***BioAPI_Load*** (8.1.9) для этого ПБУ.

6.4 Установка и удаление ПБУ и ПБФ

6.4.1 Для установки ПБУ или ПБФ используют УУИД (параметр функций ***BioAPI_Util_InstallBSP*** и ***BioAPI_Util_InstallBFP***) для идентификации ПБУ или ПБФ. Установка ПБУ или ПБФ с помощью УУИД, с которым ранее уже был установлен ПБУ или ПБФ, должна приводить к отказу установки и возврату ошибки. Если один и тот же ПБУ или ПБФ установлен в нескольких биометрических системах, он должен иметь один и тот же УУИД во всех системах.Необходимо, чтобы УУИД был уникальным для каждого ПБУ и ПБФ в отдельно взятой биометрической системе.

6.4.2 Установка ПБФ осуществляется путем вызова функции инфраструктуры БиоАПИ (10.2.1) с последующим предоставлением данных о ПБУ (схема ПБУ, определенная в 7.16).

Примечание — Данный вызов функции стандартизован, но осуществляется, как правило, с помощью установочного приложения, а не обычного биометрического приложения.

6.4.3 При установке нового ПБУ его обычно активизируют путем инсталляционного процесса, используя зависимые от реализации механизмы, а затем ПБУ может использовать механизм обратного вызова (9.2.2) для получения информации от инфраструктуры об установленных ПБФ.

Примечание — Данная функция может быть вызвана в любое время для того, чтобы ПБУ мог обновлять любую внутреннюю информацию, которую он может поддерживать. Формат и содержание (и даже существование) такой внутренней информации не стандартизированы и полностью определяются ПБУ.

6.4.4 После успешной установки или удаления ПБУ или ПБФ, следующий запрос от любого приложения или ПБУ, касающийся информации реестра компонентов, должен возвращать правильную информацию в отношении недавно установленного ПБУ или ПБФ и не возвращать информацию об удаленном ПБУ или ПБФ (8.1.4 и 8.1.10).

6.4.5 Не существует механизмов для информирования установленного ПБУ (используемого или неиспользуемого в настоящее время) об установке новых ПБФ или удалении существующих ПБФ (6.4.6). Ответственность за определение того, какие ПБФ установлены, возложена исключительно на ПБУ, использующего функцию обратного вызова перечня ПБФ (9.2.2).

6.4.6 Функция ПИП БиоАПИ ***BioAPI_Util_InstallBsp*** (10.2.1), которую вызывает приложение (например, мастер удаления), сообщает инфраструктуре БиоАПИ, что ПБУ был удален. Инфраструктура БиоАПИ обновляет реестр компонентов. Не существует стандартизованных функций для уведомления отдельных ПБУ, находящихся в рабочей памяти выполняющихся приложений, об удалении используемых ими ПБФ, и эффект удаления используемого ПБУ ПБФ зависит от исполнения.

Примечание — Удаление ПБФ обычно не вызывает удаления связанных с ним аппаратных средств или драйверов модулей БиоАПИ, так как эти драйверы и аппаратные средства могут также использоваться другими установленными ПБФ. Данная область находится вне стандартизации БиоАПИ и зависит от исполнения ПБФ.

6.5 Загрузка ПБУ и присоединение модуля БиоАПИ

6.5.1 Действия биометрического приложения должны выполняться в следующем порядке:

a) инициализация доступа к инфраструктуре БиоАПИ (8.1.1);

b) идентификация и загрузка одного или более ПБУ (8.1.4 и 8.1.5) с возможным указанием обработчика событий для получения обратных вызовов при возникновении связанных с данным ПБУ определенных событий (7.26), например подключение или удаление модуля БиоАПИ, к которому может быть осуществлен доступ через данного ПБУ. Когда обработчики события обратного вызова определены, приложение будет получать уведомления о всех подключениях и отключениях, о которых сообщается соответствующим ПБУ;

c) присоединение единственного ПБУ вместе с одним БиоАПИ каждой категории (к которому ПБУ имеет прямой или косвенный доступ).

Примечание — Под присоединением ПБУ подразумевается установление присоединенной сессии ПБУ;

d) после присоединения (со ссылкой на присоединенный ПБУ) могут быть выполнены остальные вызовы функций БиоАПИ, которые будут обработаны присоединенным ПБУ с использованием идентифицированных модулей БиоАПИ требуемой категории.

Примечание 1 — Приложение может устанавливать несколько одновременно присоединенных сессий с разными ПБУ (или с тем же ПБУ).

Примечание 2 — Пример кода последовательных вызовов приведен в приложении D.

6.5.2 Загрузка ПБУ (8.1.5) дает приложению возможность получать полную информацию о модулях БиоАПИ, к которым может быть осуществлен доступ (прямой или косвенный) через данного ПБУ или с помощью запроса инфраструктуры, или в соответствии с уведомлением обратного вызова от ПБУ (или двумя способами).

6.5.3 Модуль БиоАПИ может использоваться только в случае, если аппаратные и программные средства, от которых он зависит, в данное время подключены к системе. В модели это называется «подключенным состоянием модуля БиоАПИ».

6.5.4 Когда приложение присоединяет ПБУ, оно может указать, что модуль БиоАПИ, выбираемый для какой-либо конкретной категории модуля БиоАПИ, должен быть определен ПБУ. Это называют «выбором модуля БиоАПИ по умолчанию с использованием параметра BioAPI_DONT_CARE».

Примечание — Единственное различие между выбором конкретного доступного модуля БиоАПИ (находящегося в подключенном состоянии) данной категории и выбором модуля БиоАПИ по умолчанию с использованием параметра BioAPI_DONT_CARE заключается в том, что, в последнем случае, решение о выборе используемого модуля БиоАПИ принимает ПБУ.

6.5.5 Уведомление о событии подключения каждого модуля БиоАПИ содержит его схему.

Примечание — Возможно, что физическое подключение некоторых частей аппаратных средств (например, смарт-карта, которая может поддерживать как архивирование, так и сопоставление) будет вызывать два отдельных различных уведомления. Приложение может не связать эти два события с одним физическим устройством.

6.5.6 В ответ на функцию ***BioAPI_BSPAttach*** приложение должно выбрать не более одного модуля БиоАПИ каждой категории, который находится в настоящее время в подключенном состоянии (или выбрать BioAPI_DONT_CARE) и управляется данным ПБУ или связанным с ним ПБФ. Затем ПБУ обеспечивает доступ к модулю (для модулей БиоАПИ, управляемых напрямую) или взаимодействует со связанным с ним ПБФ для получения доступа к данному модулю БиоАПИ.

6.5.7 Информационное содержание уведомления об удалении должно включать в себя (7.28):

a) ИД модуля БиоАПИ;

b) тип события (удаление);

c) контекст обратного вызова.

6.5.8 Уведомление о событии удаления модуля БиоАПИ, который является частью набора модулей, используемых в текущей присоединенной сессии, необходимо для того, чтобы приложение больше не производило вызовов функций, кроме ***BioAPI_BSPDetach*** и ***BioAPI_GetBIRFromHandle.***

6.6 Управление модулями БиоАПИ

6.6.1 В ПИП, ИПУ и ИПФ БиоАПИ доступна функция (8.1.12 и 9.3.1.7), которая позволяет приложению управлять модулем БиоАПИ через ПБУ. ПБУ не обязательно должен поддерживать данную функцию, но если такая функция поддерживается, ПБУ идентифицирует наличие такой поддержки в своей схеме (7.16 и 7.46).

6.6.2 Данная функция содержит конкретные управляющие коды, содержание буфера и возвращаемые значения, но их формат и значение определяются конструкцией УУИД и могут быть индивидуальны для каждого изготовителя. Настоящий стандарт не устанавливает требований к УУИД, поддерживающим данную функцию.

6.7 Структура и обработка ЗБИ

6.7.1 Структура ЗБИ

ЗБИ, использующая ПИП и ИПУ, является структурой данных Си, сохраненной в памяти компьютера с использованием указателей на различные элементы. Структура ЗБИ, рекомендуемая для хранения и передачи между компьютерными системами, представляет собой сериализацию структуры данных Си и

изображена на рисунке 3. В формате, рекомендуемом для хранения и передачи, к полям, определенным в структуре данных Си, добавляются поля длины.

Примечание — Формат постоянного клиента БиоАПИ, рекомендуемый для хранения и передачи ЗБИ БиоАПИ, приведен в приложении В. Для представления ЗБИ БиоАПИ в ПИП и ИПУ используется структура данных, определенная в 7.4.

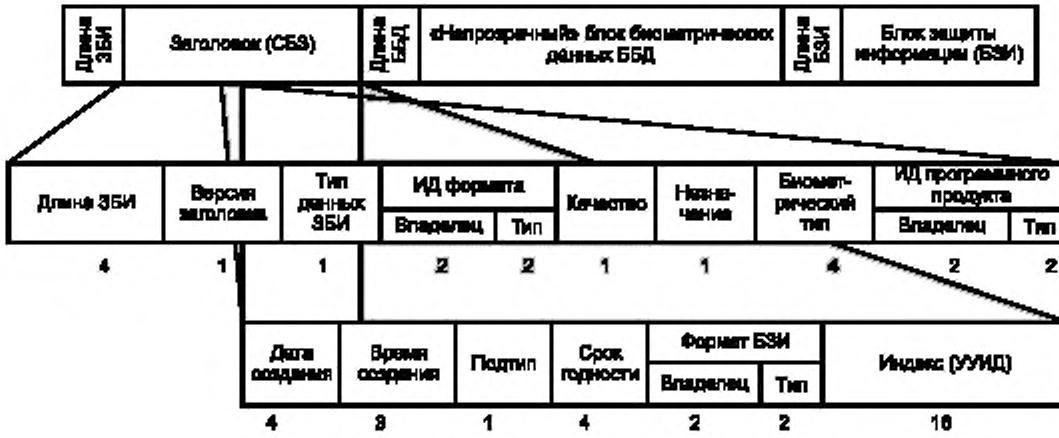


Рисунок 3 — Запись биометрической информации

СБЗ содержит информацию о содержании последующего ББД.

ББД содержит биометрический образец, формат которого определен полем «ИД формата» в СБЗ. Данный формат может быть стандартным или внутренним.

Примечание — Стандартные форматы ББД определены в ISO/IEC 19794 [5].

БЗИ является необязательным (нулевое поле длины БЗИ указывает на отсутствие данного блока) и содержит параметры, связанные с использованием цифровой подписи и/или шифрованием ЗБИ. Формат БЗИ определяется полем «Формат БЗИ» в СБЗ. Поле «Тип данных ЗБИ» указывает, использует ли ЗБИ цифровую подпись и/или является ли ББД зашифрованным (7.9).

Первое поле длины указывает длину всей ЗБИ без учета самого поля. Второе поле длины, расположенное после заголовка и перед ББД, указывает длину ББД без учета самого поля длины. Третье поле длины, расположенное после ББД и перед БЗИ, указывает длину БЗИ без учета самого поля длины.

6.7.2 Обработка данных ЗБИ

При создании новой ЗБИ, ПБУ возвращает к ней дескриптор. Большинство локальных операций может быть выполнено без перемещения ЗБИ из ПБУ. (Так как ЗБИ могут быть достаточно большими, это является преимуществом при выполнении различных операций). Однако если приложению необходимо управлять ЗБИ (сохранить ее в базе данных приложения, передать другому ПБУ или переслать на сервер для верификации/идентификации, возможно, через инфраструктуру с использованием ISO/IEC 24708 [6]), оно может запрашивать ЗБИ с использованием дескриптора (функции BioAPI_GetBIRFromHandle). Если требуется только информация заголовка, она может быть получена с использованием функции BioAPI_GetHeaderFromHandle.

Передача ЗБИ в качестве входного параметра функции БиоАПИ может быть реализована одним из следующих трех способов:

a) ссылкой на ее дескриптор (если она находится в вызванном ПБУ);

b) ссылкой на ее ключевое значение (УУИД) в открытой базе данных ЗБИ (управляемой вызванным ПБУ);

c) предоставлением самой ЗБИ с использованием структуры данных BioAPI_BIR Си.

7 Типы и макросы БиоАПИ

7.1 Макрос БиоАПИ

Определение соглашения о вызовах БиоАПИ:

```
#ifdef (WIN32)
#define BioAPI__stdcall
#else
#define BioAPI
#endif
```

Примечание — Взаимодействие программных продуктов разных изготовителей в общем случае зависит от выбранного (иногда в соответствии с установками в заголовке Си) представления в памяти (например, заполнение промежутков между элементами данных, механизм передачи параметров и использование регистров или стеков). Многие операционные системы выбирают один из вариантов по умолчанию, который и должен использоваться. Если не существует выбранных установок по умолчанию, необходимо использовать опцию представления структур данных без заполнения промежутков между элементами данных.

7.2 Тип BioAPI_BFP_LIST_ELEMENT

7.2.1 Данный тип определяет ПБФ, представляя его категорию и УУИД. При запросе о поддерживаемых ПБФ, ПБУ возвращается список.

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

7.2.2 Определения

BFPCategory — определяет категорию модуля БиоАПИ, поддерживаемую ПБФ.

BFPUuid — УУИД ПБФ в реестре компонентов.

7.3 Тип BioAPI_BFP_SCHEMA

7.3.1 Информация о ПБФ, содержащаяся в реестре компонентов.

```
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;
```

7.3.2 Определения

BFPUuid — УУИД ПБФ.

BFPCategory — определяет категорию модуля БиоАПИ, поддерживаемую ПБФ.

BFPDescription — строка с нулевым символом на конце, содержащая текстовое описание ПБФ.

Path — указатель на строку с нулевым символом на конце, содержащую путь к файлу ПБФ, включая название файла. Путь к файлу может быть записан в виде адреса страницы URL. Символьная строка должна содержать символы, закодированные в формате UTF-8 в соответствии с ИСО/МЭК 10646.

Примечание — Если в вызове функции используется BioAPI_BFP_SCHEMA, принимающий компонент выделяет память для элемента схемы *Path*, а вызывающий компонент освобождает память.

SpecVersion — номер редакции и номер поправки или изменений редакции стандарта, в соответствии с которой был разработан ПБФ.

Примечание — Требования к ПБФ будут установлены в следующих частях комплекса стандартов 19784.

ProductVersion — строка версии программного обеспечения ПБФ.

Vendor — строка с нулевым символом на конце, содержащая название изготовителя ПБФ.

BFPSupportedFormats — указатель на массив структур BioAPI_BIR_BIOMETRIC_DATA_FORMAT, определяющий поддерживаемые форматы ББД.

NumSupportedFormats — число поддерживаемых форматов, содержащихся в BFPSupportedFormats.

FactorsMask — маска, указывающая, какие биометрические типы поддерживаются ПБФ.

BFPPropertyID — УУИД формата передаваемого свойства ПБФ.

BFPProperty — адрес и длина буфера памяти, содержащего свойства ПБФ. Формат и содержание свойства ПБФ могут быть определены изготовителем или установлены в соответствующем стандарте.

7.4 Тип BioAPI_BIR

7.4.1. BioAPI_BIR представляет собой контейнер биометрических данных и состоит из BioAPI_BIR_HEADER, ББД и (необязательно) БЗИ. ББД может содержать исходные данные образца, частично обработанные (промежуточные) данные или полностью обработанные биометрические данные. BioAPI_BIR может использоваться для регистрации пользователя (при этом он хранится постоянно) или для верификации или идентификации пользователя (при этом он используется кратковременно).

7.4.2 ББД и БЗИ представляют собой целое число октетов и имеют переменную длину до 2**32-1 октетов. Если БЗИ содержит цифровую подпись, она вычисляется одновременно для BioAPI_BIR_Header и ББД.

```
typedef struct bioapi_bir {
        BioAPI_BIR_HEADER Header;
        BioAPI_DATA BiometricData;
        BioAPI_DATA SecurityBlock;  /* Если БЗИ отсутствует, то SecurityBlock.Data=NULL */
} BioAPI_BIR;
```

Примечание 1 — ЗБИ БиоАПИ содержит информацию, необходимую для формата постоянного клиента ЕСФОБД определенную в ИСО/МЭК 19785-1.

Примечание 2 — Форматы BiometricData и SecurityBlock определены соответственно элементами BioAPI_BIR_BIOMETRIC_DATA_FORMAT и BioAPI_BIR_SECURITY_BLOCK_FORMAT в заголовке BioAPI_BIR_HEADER.

Примечание 3 — ЕСФОБД позволяет использовать форматы ЗБИ, отличающиеся от формата, поддерживаемого БиоАПИ. Преобразование между форматом ЗБИ БиоАПИ и другими форматами ЗБИ определено в ИСО/МЭК 19785-1.

7.5 Тип BioAPI_BIR_ARRAY_POPULATION

Данный тип представляет собой массив ЗБИ, используемый при идентификации (в качестве входного параметра для функций *BioAPI_Identify* или *BioAPI_IdentifyMatch*, как часть BioAPI_IDENTIFY_POPULATION).

```
typedef struct bioapi_bir_array_population {
        uint32_t NumberOfMembers;
        BioAPI_BIR *Members; /* указатель на массив ЗБИ */
} BioAPI_BIR_ARRAY_POPULATION;
```

7.6 Тип BioAPI_BIR_BIOMETRIC_DATA_FORMAT

Данный тип определяет формат данных, содержащихся в элементе БиоАПИ BiometricData «непрозрачного» блока биометрических данных (ББД) ЗБИ.

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

Примечание 1 — Значения FormatOwner (владелец формата) присваиваются и регистрируются органами регистрации ЕСФОБД. Значения FormatType (тип формата) присваиваются владельцем формата и при необходимости тип формата может быть зарегистрирован. Информация о регистрации приведена в ИСО/МЭК 19785-2.

Примечание 2 — BioAPI_BIR_BIOMETRIC_DATA_FORMAT соответствует комбинации «CBEFF_BDB_ _format_owner» и «CBEFF_BDB_format_type», установленных в ИСО/МЭК 19785-1.

Примечание 3 — Данная структура используется обычно в заголовке ЗБИ, однако она также используется как входной параметр для функций, осуществляющих получение биометрических данных.

7.7 Тип BioAPI_BIR_BIOMETRIC_PRODUCT_ID

Данный тип предоставляет идентификатор продукта (ИДП) для ПБУ, который разработал ББД в ЗБИ (элемент BiometricData).

```
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)
```

Значение NO_VALUE_AVAILABLE следует отражать установкой значений всех компонентов в ноль. Это значение следует использовать только для ЗБИ, которая первоначально была получена не от ПБУ БиоАПИ, а из другого источника, и была преобразована в ЗБИ БиоАПИ. БПУ не должны использовать это значение.

Значения ProductOwner (владелец продукта) присваиваются и регистрируются органами регистрации ЕСФОБД в качестве идентификаторов биометрической организации. Информация об органах регистрации приведена в ИСО/МЭК 19785-2. Тип продукта присваивается владельцем продукта и может быть зарегистрирован.

Примечание 1 — ИД продуктов аналогичны ИД форматов и процессы их регистрации идентичны. Изготовитель может зарегистрировать значение владельца формата/продукта (идентификатор биометрической организации), которое может использоваться в обоих полях.

Примечание 2 — BioAPI_BIR_BIOMETRIC_PRODUCT_ID соответствует «CBEFF_BDB_product_owner» и «CBEFF_BDB_product_type», приведенным в ИСО/МЭК 19785-1.

7.8 Тип BioAPI_BIR_BIOMETRIC_TYPE

Данный тип представляет собой маску, описывающую набор биометрических типов (факторов), содержащихся в рамках ЗБИ БиоАПИ или поддерживаемых ПБУ.

```
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)
#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)
```

Примечание 1 — BioAPI_TYPE_MULTIPLE используется для обозначения того, что биометрические образцы, содержащиеся в ББД (BiometricData ЗБИ), включают в себя образцы, полученные от биометрических сканеров разных типов (например, данные отпечатков пальцев и изображения лица). Расположение индивидуальных образцов в ББД определяет владелец формата и идентифицируется значением типа формата.

Примечание 2 — Значение NO_VALUE_AVAILABLE указывается установкой нулевого значения. Данное значение должно использоваться в том случае, если для ЗБИ, которые первоначально не были созданы ПБУ БиоАПИ, а были преобразованы в ЗБИ БиоАПИ, информация о биометрическом типе недоступна в записи первоначального источника. Преобразованные ЗБИ, чьи биометрические типы не соответствуют ни одному из определенных типов, должны использовать значение BioAPI_TYPE_OTHER.

Примечание 3 — BioAPI_BIR_BIOMETRIC_TYPE соответствует «CBEFF_BDB_biometric_type» по ИСО/МЭК 19785-1.

7.9 Тип BioAPI_BIR_DATA_TYPE

7.9.1 BioAPI_BIR_DATA_TYPE (тип данных ЗБИ БиоАПИ) используется для решения следующих задач:

a) определения типа биометрических образцов (исходные, промежуточные или обработанные), которые содержатся в ББД;

b) определения того, зашифрована ли ЗБИ и/или использует ли она цифровую подпись;

c) определения того, включено или нет значение индекса в качестве составной части заголовка ЗБИ.

Примечание — Если ЗБИ зашифрована ПБУ, то она может не определяться приложением или другим ПБУ.

7.9.2 Должен быть установлен один из следующих трех признаков: «исходный» (RAW), «промежуточный» (INTERMEDIATE) или «обработанный» (PROCESSED). Если ЗБИ, содержащая данные ЗБИ с установленными различными признаками, передается в инфраструктуру БиоАПИ в качестве параметра вызываемой функции, должно возвращаться значение ошибки BioAPIERR_INVALID_BIR.

Примечание — ЗБИ, которые первоначально не были созданы ПБУ БиоАПИ, а были преобразованы из другого формата данных и для которых информация о типе образца недоступна, могут не устанавливать данный признак (ПБУ БиоАПИ должны устанавливать один из вышеуказанных признаков).

7.9.3 Установка признаков «зашифровано» (ENCRYPTED) и «подписано» (SIGNED) является необязательной.

7.9.4 Признак «индекс» (INDEX_PRESENT) следует устанавливать в случае, если индекс присутствует в заголовке ЗБИ, и не следует устанавливать в случае, если индекс отсутствует в заголовке ЗБИ.

```
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_ENCRYPTED              (0x10)
        #define BioAPI_BIR_DATA_TYPE_SIGNED                 (0x20)
        #define BioAPI_BIR_INDEX_PRESENT                    (0x80)
```

Примечание — BioAPI_BIR_DATA_TYPE соответствует комбинации «CBEFF_BDB_processed_level» и «CBEFF_BIR_integrity_options» по ИСО/МЭК 19785-1.

7.10 Тип BioAPI_BIR_HANDLE

Данный тип является дескриптором для обращения к ЗБИ БиоАПИ, существующей в ПБУ.

Примечание — Дескриптор, определяющий ЗБИ, имеет положительное ненулевое значение. Другие значения BioAPI_BIR_HANDLE (в настоящее время только минус 1 и минус 2) зарезервированы для индикации исключений.

```
typedef int32_t BioAPI_BIR_HANDLE;
        #define BioAPI_INVALID_BIR_HANDLE                   (-1)
        #define BioAPI_UNSUPPORTED_BIR_HANDLE               (-2)
```

7.11 Тип BioAPI_BIR_HEADER

Данный тип представляет собой заголовок ЗБИ, содержащий стандартную информацию, с описанием содержания следующих за ней непрозрачных ББД. Данная информация может быть прочитана приложением и предоставляется для того, чтобы обеспечить приложению возможность выбора метода обработки и трассировки с учетом ЗБИ. ПБУ не зашифровывает данный заголовок.

```
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;
```

Примечание 1 — BioAPI_BIR_HEADER соответствует СБЗ в ЕСФОБД по ИСО/МЭК 19785-1.

Примечание 2 — Срок годности (Expiration date) соответствует элементу «Valid to» поля «CBEFF_BDB_validity_period» по ИСО/МЭК 19785-1. Поле признака (Index) соответствует полю «CBEFF_BDB_index» по ИСО/МЭК 19785-1.

Примечание 3 — Возможно существование ЗБИ БиоАПИ, которая не была создана ПБУ, а была преобразована из другого формата данных. В этом случае некоторые из полей заголовка, которые являются

необязательными в ЕСФОБД (ИСО/МЭК 19785-1), но требуются БиоАПИ, могут отсутствовать. В этом случае для данных полей предусмотрено значение NO_VALUE_AVAILABLE или значение по умолчанию (в соответствующих этим полям структурах данных). Однако все ЗБИ, созданные ПБУ БиоАПИ, должны содержать корректные данные для этих полей и не должны использовать значение NO_VALUE_AVAILABLE (исключение составляют поля BioAPI_Quality и BioAPI_BIR_SUBTYPE, которые являются необязательным в заголовке ЗБИ БиоАПИ). Если ЗБИ, созданная не БиоАПИ, обозначена как входной параметр для ПБУ, ПБУ может возвратить ошибку «недействительная ЗБИ».

Примечание 4 — Формат хранения ЗБИ включает в себя точную длину поля, которая не является необходимой в структуре Си (приложение В относительно формата хранения ЗБИ).

7.12 Тип BioAPI_BIR_PURPOSE

7.12.1 Данный тип определяет назначение создаваемой ЗБИ БиоАПИ (при использовании в качестве входного параметра к функциям БиоАПИ) или назначение имеющейся ЗБИ (при использовании в качестве выходного параметра функции БиоАПИ или в заголовке ЗБИ).

```
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_NO_PURPOSE_AVAILABLE                             (0)
```

Примечание — Условие BioAPI_NO_DATA_AVAILABLE (данные недоступны) указывается установкой значения в ноль. Данное значение используется только для тех ЗБИ, которые первоначально не сформированы ПБУ БиоАПИ, а были созданы другим источником и преобразованы в ЗБИ БиоАПИ. ПБУ не следует использовать данное значение.

7.12.2 Назначение ЗБИ (BioAPI_BIR_PURPOSE) используется двумя способами. Во-первых, оно используется как входной параметр, чтобы дать возможность приложению указать ПБУ назначение итоговой ЗБИ, таким образом, давая возможность ПБУ выполнить соответствующую регистрацию и обработку, чтобы создать надлежащую ЗБИ для данного назначения. Во-вторых, оно используется в пределах заголовка ЗБИ, чтобы указать приложению (или ПБУ в течение последующих операций), какому назначению соответствует ЗБИ. Например, некоторые ПБУ используют различные форматы ББД в зависимости от их использования для верификации или идентификации; в последнем случае формат обычно включает в себя дополнительную информацию для увеличения скорости или точности. Многие ПБУ используют различные форматы данных в зависимости от их использования в качестве образца для непосредственной верификации или в качестве контрольного шаблона для будущих сопоставлений (при регистрации).

Примечание — Параметр BioAPI_BIR_PURPOSE в заголовке ЗБИ соответствует параметру CBEFF_BDB_purpose по ИСО/МЭК 19785-1. Названия параметров отличаются незначительно, так как ЗБИ БиоАПИ ограничена отдельной BDB, но семантика остается неизменной.

7.12.3 Ограничения на использование данных, содержащихся в ЗБИ конкретного назначения:

a) в заголовке ЗБИ может быть указано любое назначение;

b) назначения BioAPI_PURPOSE_VERIFY (верификация) и BioAPI_PURPOSE_IDENTIFY (идентификация) допустимы только в качестве входного параметра ***функции BioAPI_Capture;***

c) назначения BioAPI_PURPOSE_ENROLL (регистрация), BioAPI_PURPOSE_ENROLL_FOR_ _VERIFICATION_ONLY (регистрация только для верификации) и BioAPI_PURPOSE_ENROLL_FOR_ _IDENTIFICATION_ONLY (регистрация только для идентификации) допустимы только в качестве входных параметров функций ***BioAPI_Capture, BioAPI_Enroll*** и ***BioAPI_Import*** (импортирование);

d) назначение BioAPI_PURPOSE_AUDIT (контроль) не является входным параметром функции, а используется только в заголовке ЗБИ;

e) функции ***BioAPI_Process*** (обработка), ***BioAPI_CreateTemplate*** (создать шаблон) и ***BioAPI_ProcessWithAuxData*** не используют назначение в качестве входного параметра, а считывают поле назначения из заголовка входного ЗБИ *CapturedBIR* (полученная ЗБИ);

f) функция ***BioAPI_Process*** (обработка) может принимать в качестве входных данных любую промежуточную ЗБИ с назначением BioAPI_PURPOSE_VERIFY и BioAPI_PURPOSE_IDENTIFY и должна возвращать ЗБИ с тем же назначением, что и входная ЗБИ;

g) функция ***BioAPI_CreateTemplate*** (создать шаблон) может принимать в качестве входных данных любую промежуточную ЗБИ с назначением: BioAPI_PURPOSE_ENROLL, BIOAPI_PURPOSE_ENROLL_FOR_

_VERIFICATION_ONLY, BIOAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY и должна возвращать ЗБИ с тем же назначением, что и входная ЗБИ;

h) если ЗБИ подходит для регистрации, верификации и идентификации, то возвращаемая ЗБИ должна иметь назначение BioAPI_PURPOSE_ENROLL.

7.13 Тип BioAPI_BIR_SECURITY_BLOCK_FORMAT

Данный тип определяет формат данных, содержащихся в блоке безопасности (БЗИ) ЗБИ БиоАПИ (элемент блока безопасности).

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

Если признаки «шифрование» и «подпись» не установлены в поле Bio API_BIR_DATA_TYPE заголовка ЗБИ, то параметры SecurityFormatOwner и SecurityFormatType должны быть установлены в 0x0000 и блок безопасности должен отсутствовать.

Значения SecurityFormatOwner присвоены и зарегистрированы регистрационными правами ЕСФОБД как идентификаторы биометрических организаций по ИСО/МЭК 19785-2. SecurityFormatType присвоен владельцем формата безопастности (биометрическая организация) и может быть зарегистрирован дополнительно.

Примечание 1 — ИД форматов безопасности аналогичны ИД форматов. Процесс регистрации ИД форматов безопасности аналогичен процессу регистрации ИД форматов. Изготовитель может зарегистрировать значение владельца формата/продукта/безопасности (идентификатор биометрической организации), что может быть использовано во всех связанных областях.

Примечание 2 — Содержание блока безопасности может включать в себя цифровую подпись или код аутентификационного сообщения (вычисленный как заголовок ЗБИ + ББД), параметры шифрования ББД (то есть алгоритмы шифрования, длина ключа) и/или целостные параметры ЗБИ (ИД алгоритмов, название ключа, версия).

Примечание 3 — Формат блока безопасности ЗБИ БиоАПИ в заголовке ЗБИ соответствует «CBEFF_SB_format_owner» и «CBEFF_SB_format_type» по ИСО/МЭК 19785-1.

7.14 Тип BioAPI_BIR_SUBTYPE

7.14.1 Данный тип идентифицирует подтип в рамках типа ББД указанного в BioAPI_BIR_BIOMETRIC_ _TYPE. Значения подтипов определяются спецификациями данного типа ББД.

7.14.2 Каждый из флагов BioAPI_BIR_SUBTYPE_LEFT и BioAPI_BIR_SUBTYPE_RIGHT (ноль, один или оба) может быть установлен произвольно.

7.14.3 Может быть установлен один или ни одного из пяти подтипов пальца.

```
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)
```

Примечание 1 — Условие NO_VALUE_AVAILABLE (значение недоступно) указывается присвоением значения ноль. Для ЗБИ, которые не были первоначально созданы ПБУ БиоАПИ, а были преобразованы из другого формата данных и для которых информация о подтипе недоступна, допускается данное значение не устанавливать.

Примечание 2 — BioAPI_BIR_SUBTYPE соответствуют «CBEFF_BDB_biometric_subtype» по ИСО/МЭК 19785-1.

Примечание 3 — Данная структура прежде всего используется в заголовке ЗБИ, однако она также используется как входной параметр для функций получения биометрических данных. Значение BioAPI_NO_SUBTYPE_AVAILABLE используется в заголовке ЗБИ, если данное поле неприменимо или информация недоступна. Значение BioAPI_NO_SUBTYPE_AVAILABLE также используется как параметр функции, если приложение позволяет ПБУ определить, какой тип данных будет получен.

7.15 Тип BioAPI_BOOL

Данный тип используется для индикации истинного или ложного значения условия.

```
typedef uint8_t BioAPI_BOOL;
        #define BioAPI_FALSE                    (0)
        #define BioAPI_TRUE                     (!BioAPI_FALSE)
```

7.16 Тип BioAPI_BSP_SCHEMA

7.16.1 Данный тип включает в себя информацию о ПБУ, содержащуюся в реестре компонентов БиоАПИ.

```
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;
} BioAPI_BSP_SCHEMA;
```

7.16.2 Определения

BSPUuid — УУИД ПБУ.

BSPDescription — строка с нулевым символом на конце, содержащая текстовое описание ПБУ.

Path — указатель на строку с нулевым символом на конце, содержащую путь к файлу ПБФ, включая название файла. Путь к файлу может быть записан в виде адреса страницы URL. Символьная строка должна содержать символы, закодированные в формате UTF-8 в соответствии с ИСО/МЭК 10646.

Примечание — Если BioAPI_BSP_SCHEMA используется в вызове функции, компонент, получающий вызов, выделяет память для элемента схемы Path (путь к файлу), а вызывающий компонент освобождает память.

SpecVersion — номер редакции и номер поправки или изменений данной редакции спецификации БиоАПИ, для которой был разработан ПБУ.

ProductVersion — строка версии программного обеспечения ПБУ.

Vendor — строка с нулевым символом на конце, содержащая название изготовителя ПБУ.

BSPSupportedFormats — указатель на структуру BioAPI_BIR_BIOMETRIC_DATA_FORMAT, определяющую поддерживаемые форматы ББД.

NumSupportedFormats — число поддерживаемых форматов, содержащихся в BspSupportedFormats.

FactorMask — маска, указывающая биометрические типы, поддерживаемые ПБУ.

Operations — маска, указывающая операции, поддерживаемые ПБУ.

Options — маска, указывающая опции, поддерживаемые ПБУ.

PayloadPolicy — пороговое значение (минимальное значение ОЛС), используемое для принятия решения о выдаче полезной информации после успешной верификации.

MaxPayloadSize — максимальный размер полезной информации (в байтах), которую может принять ПБУ.

DefaultVerifyTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для функции верификации ***BioAPI_Verify*** в случае, когда время ожидания не определено приложением.

DefaultIdentifyTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для функции идентификации ***BioAPI_Identify*** и ***BioAPI_IdentifyMatch*** в случае, когда время ожидания не определено приложением.

DefaultCaptureTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для функции захвата ***BioAPI_Capture*** в случае, когда время ожидания не определено приложением.

DefaultEnrollTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для функции регистрации ***BioAPI_Enroll*** в случае, когда время ожидания не определено приложением.

DefaultCalibrateTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для операций калибровки датчика в случае, когда время ожидания не определено приложением.

MaxBSPDbSize — максимальный размер управляемой ПБУ базы данных ЗБИ.

Примечание 1 — Применяется только в том случае, если ПБУ способен непосредственно управлять отдельным модулем архива.

Примечание 2 — Нулевое значение означает, что информация о размере базы данных не должна быть предоставлена по следующим трем причинам:

a) база данных не поддерживается;

b) существует возможность управления несколькими модулями (непосредственно или с использованием интерфейса ПБФ), каждый из которых может иметь различную максимальную длину и информация о данном модуле будет предоставлена как часть уведомления о подключении (часть схемы модуля);

c) поддерживается один модуль архива, но в данном параметре информация не предоставлена — она будет доступна в уведомлении о подключении.

MaxIdentify — максимальное число людей, поддерживаемых функцией идентификации. Значение FFFFFFFF указывает на отсутствие ограничения.

7.16.3 Более подробное описание приведенных элементов и порядка записи информации ПБУ в реестр компонентов БиоАПИ приведено в 10.1.2 и 10.2.1.

7.17 Тип BioAPI_CANDIDATE

Данный тип содержит одного кандидата из набора, возвращаемого функциями идентификации ***BioAPI_Identify*** или сопоставления ***BioAPI_IdentifyMatch*** и соответствующий успешному сопоставлению.

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

7.18 Тип BioAPI_CATEGORY

Данное битовое поле описывает категорию ПБФ или модуля БиоАПИ. ПБФ или модуль БиоАПИ должен соответствовать только одной категории.

```
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)
```

Старший значимый бит зарезервирован и не указывает категорию ПБФ или модуля БиоАПИ.

7.19 Тип BioAPI_DATA

7.19.1 Структура BioAPI_DATA используется для связывания длины (в байтах) с адресом произвольного блока непрерывной памяти.

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

7.19.2 Определения

Length — длина буфера данных в байтах.

Data — указатель на начало буфера данных произвольной длины.

7.20 Тип BioAPI_DATE

Данный тип определяет дату создания или срок действия ЗБИ.

```
typedef struct bioapi_date {
        uint16_t Year; /*диапазон значений: 1900 — 9999 */
        uint8_t Month; /*диапазон значений: 01 — 12 */
        uint8_t Day; /* диапазон значений: 01 — 31, согласуется с Month, Year */
} BioAPI_DATE;
 #define BioAPI_NO_YEAR_AVAILABLE (0)
 #define BioAPI_NO_MONTH_AVAILABLE (0)
 #define BioAPI_NO_DAY_AVAILABLE (0)
```

Условие BioAPI_NO_DATA_AVAILABLE (данные недоступны) указывается установкой нулевого значения. Если информация о дате, используемая в поле год — день — время (ГДВ) в заголовке ЗБИ, недоступна, то должно быть использовано значение NO_DATE_AVAILABLE.

Примечание 1 — 2000-й год представлен значением 2000.

Примечание 2 — При использовании ГДВ в качестве ExpirationDate в заголовке ЗБИ соответствует части «срок действия» параметра «CBEFF_BDB_validity_period» по ИСО/МЭК 19785-1.

Примечание 3 — Форматы даты соответствуют установленным в ИСО 8601 [2].

7.21 Тип BioAPI_DB_ACCESS_TYPE

Данное битовое поле описывает необходимый уровень доступа биометрического приложения к управляемой ПБУ базе данных ЗБИ. ПБУ может использовать эту маску, чтобы определить, какую блокировку базы данных ЗБИ он должен обеспечить.

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

7.22 Тип BioAPI_DB_MARKER_HANDLE

Данный тип представляет собой дескриптор к маркеру базы данных ЗБИ.

Маркер является внутренней структурой данных, управляемой ПБУ, которая динамически указывает на запись в открытой базе данных ЗБИ, управляемой ПБУ. Дескриптор маркера создается и возвращается биометрическому приложению функциями ***BioAPI_DbOpen*** и ***BioAPI_DbGetBIR.*** Маркер содержит дескриптор открытой базы данных и позицию записи в этой базе. Все маркеры (и их дескрипторы) к открытой базе данных ЗБИ, хранимые биометрическим приложением, становятся недоступными при закрытии биометрическим приложением базы данных.

```
typedef uint32_t BioAPI_DB_MARKER_HANDLE;
```

7.23 Тип BioAPI_DB_HANDLE

Данный тип представляет собой дескриптор к открытой базе данных ЗБИ, первоначально передаваемый ПБУ биометрическому приложению, а затем используемый данным приложением для обращения к базе данных через ПБУ.

```
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)
```

7.24 Тип BioAPI_DBBIR_ID

Данный тип определяет структуру, содержащую дескриптор базы данных ЗБИ, управляемой ПБУ и УУИД ЗБИ в данной базе данных.

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

Примечание — Данный тип используется как элемент параметра BioAPI_INPUT_BIR.

7.25 Тип BioAPI_DTG

Данный тип определяет дату и время создания ЗБИ.

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

Примечание — ГДВ БиоАПИ в заголовке ЗБИ соответствует CBEFF_creation_date по ИСО / МЭК 19785-1.

7.26 Тип BioAPI_EVENT

Данный перечислимый тип определяет виды событий, которые могут быть вызваны ПБУ, ПБФ или модулем БиоАПИ. Биометрическое приложение может определять функции обратного вызова обработчика событий типа BioAPI_EventHandler для получения и управления данными событиями. Функции обратного вызова регистрируются функцией ***BioAPI_BSPLoad.*** Например события, включающие в себя добавление (подключение) или удаление биометрического датчика. События являются асинхронными.

События BioAPI_NOTIFY_SOURCE_PRESEN и BioAPI_NOTIFY_SOURCE_REMOVED формируются устройством (модулем датчиков), которое способно определить доступен ли пользователь для предоставления биометрического образца (например, находится ли палец на сканере). BioAPI_NOTIFY_SOURCE_PRESENT показывает, что образец доступен, в то время как BioAPI_NOTIFY_SOURCE_REMOVED показывает, что образец более недоступен. Не требуется, чтобы данные события появлялись парами; несколько событий BioAPI_NOTIFY_SOURCE_PRESENT могут появляться последовательно.

```
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)
```

7.27 Тип BioAPI_EVENT_MASK

Данный тип определяет маску с позициями битов для каждого типа событий. Маска используется для разрешения и запрещения выдачи уведомлений о событиях и для указания, какие события поддерживаются ПБУ.

```
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)
```

Примечание — Иногда невозможно определить в маске событие подключения, возникающее в прикрепленной сессии ПБУ, из-за того, что событие может появиться после вызова функции ***BioAPI_BSPLoad*** и перед тем как может быть обработан любой вызов ***BioAPI_EnableEvents***. Данная ситуация возникает из-за того, что для вызова функции ***BioAPI_EnableEvents*** требуется дескриптор, который возвращается функцией ***BioAPI_BSPAttach,*** а вызов функции ***BioAPI_BSPAttach*** должен следовать за вызовом функции ***BioAPI_BSPLoad.*** Событие подключения (INSERT) возникнет в ПБУ при вызове функции ***BioAPI_BSPLoad*** в случае, если модуль БиоАПИ уже «подключен», и это событие передастся приложению раньше, чем оно сможет вызвать функцию ***BioAPI_EnableEvents***.

7.28 Тип BioAPI_EventHandler

7.28.1 Данный тип представляет собой интерфейс обработчика событий, который должен поддерживать приложение, если необходимо получать асинхронные уведомления о событиях, таких как наличие биометрического образца или ошибка, обнаруженная ПБУ. Процесс обработки события регистрируется в инфраструктуре БиоАПИ в ходе выполнения функции ***BioAPI_BSPLoad.*** Данный обработчик события предназначен для всех общих событий, возникающих в загруженном ПБУ, во всех присоединенных сессиях. Уведомления об общих событиях обрабатываются через инфраструктуру БиоАПИ.

Функция обработки событий и любые функции, запрашивающие данную функцию (непосредственно или косвенно), не должны приводить к вызову функций БиоАПИ. Подобные циклические вызовы могут привести к их взаимной блокировке во многих ситуациях, поэтому обработчик события должен быть реали-

зован без использования сервисов БиоАПИ (но функции получения списков могут использоваться в любое время).

BioAPI_EventHandler может быть вызван несколько раз в ответ на единственное событие, возникшее в присоединенном ПБУ. Обработчик события и вызывающее приложение должны прослеживать получение уведомлений о событии и игнорировать дублирующие уведомления. Уведомление о событии представляет собой следующую информацию:

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

7.28.2 Определения

BSPUuid — УУИД ПБУ, инициирующий событие.

UnitID — ИД модуля БиоАПИ, связанного с событием.

AppNotifyCallbackCtx — общий указатель на контекстную информацию, которая предоставляется при вызове функции ***BioAPI_BSPLoad,*** устанавливающей обработчик событий.

UnitSchema — указатель на схему модуля БиоАПИ, связанного с данным событием.

EventType — тип произошедшего события BioAPI_ EVENT.

Если параметр *EventType* является BioAPI_NOTIFY_INSERT, то должна быть предоставлена схема модуля (параметр *UnitSchema* должен указывать на переменную типа BioAPI_UNIT_SCHEMA). В других случаях *UnitSchema* должен быть пустым.

Когда приложение получает вызов к обработчику событий, который хранит схему модуля, приложение не должно вызывать BioAPI_Free для освобождения блока памяти, содержащего схему модуля или блока памяти, указывающего на нее любым своим элементом.

7.29 Тип BioAPI_FMR

32-разрядное целочисленное значение (N), которое указывает на вероятность ошибки ложного совпадения, равную $N/(2^{31} - 1)$. Чем больше данное значение, тем больше вероятность ошибки ложного совпадения. Отрицательные значения используются в особых случаях. В настоящее время определено единственное отрицательное значение — минус 1.

```
typedef int32_t BioAPI_FMR;
    #define BioAPI_NOT_SET          (–1)
```

Примечание — ОЛС используется в БиоАПИ в качестве средства для установки порогов и возврата оценок совпадения (см. приложение С, раздел С.4).

7.30 Тип BioAPI_FRAMEWORK_SCHEMA

7.30.1 Данный тип представляет собой схему структуры компонента инфраструктуры БиоАПИ для занесения компонентов БиоАПИ в реестр.

```
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;
```

7.30.2 Описание

FrameworkUuid — УУИД компонента инфраструктуры.

FwDescription — строка с нулевым символом на конце, содержащая текстовое описание инфраструктуры.

Path — указатель на строку с нулевым символом на конце, содержащую путь и имя файла с исполняемым кодом инфраструктуры. Путь к файлу может быть записан в виде адреса страницы (URL). Данная

строка должна состоять из символов по ИСО/МЭК 10646 кодировки UTF-8 (ИСО/МЭК 10646, приложение D).

Примечание — Если в вызове функции используется BioAPI_BFP_SCHEMA, принимающий компонент выделяет память для элемента схемы *Path*, а вызывающий компонент освобождает память.

SpecVersion — номер редакции и номер поправки или изменений редакции стандарта, в соответствии с которой был разработан ПБФ.

ProductVersion — строка версии программного обеспечения инфраструктуры.

Vendor — строка с нулевым символом на конце, содержащая название изготовителя инфраструктуры.

FwPropertyID — УУИД формата следующих свойств инфраструктуры.

FwProperty — адрес и длина буфера памяти, содержащего свойства инфраструктуры. Формат и содержание свойств инфраструктуры могут быть описаны изготовителем или указаны в стандарте.

7.30.3 Определение схемы структуры приведено в 10.1.1.

7.31 Тип BioAPI_GUI_BITMAP

7.31.1 Данная структура предоставляет собой графическое изображение для его отображения приложением.

```
typedef struct bioapi_gui_bitmap {
        uint32_t Width; /* Ширина изображения в точках (число точек в горизонтальной линии) */
        uint32_t Height; /* Высота изображения в точках (число точек в вертикальной линии) */
        BioAPI_DATA Bitmap,
} BioAPI_GUI_BITMAP;
```

7.31.2 Определения

Bitmap — множество 8-битовых полутоновых точек (где «00» — черный и «FF» — белый), в котором данные считываются слева направо и сверху вниз следующим образом (таблица 1).

Таблица 1 — Формат битовой карты графического интерфейса пользователя

Позиция байта	Значение	Описание
0	Строка 0, пиксель 0	Первый пиксель первой строки
1	Строка 0, пиксель 1	Второй пиксель первой строки
...	...	
Ширина — 1	Строка 0, пиксель (ширина — 1)	Последний пиксель первой строки
Ширина	Строка 1, пиксель 0	Первый пиксель второй строки
...	...	
(Ширина х высоту) — 1	Строка (ширина — 1), пиксель (высота — 1)	Последняя строка, последний пиксель

Примечание — Данная структура используется вместе с опцией графического интерфейса пользователя, управляемой приложением. Описание BioAPI_GUI_STATE_CALLBACK и BioAPI_GUI_STREAMING_CALLBACK приведено в 7.36 и 7.37.

7.32 Тип BioAPI_GUI_MESSAGE

Данная структура предоставляет сообщение для его отображения приложением.

```
typedef uint32_t BioAPI_GUI_MESSAGE;
```

Примечание — Данная структура используется вместе с опцией графического интерфейса пользователя, управляемой приложением. Описание BioAPI_GUI_STATE_CALLBACK приведено в 7.36.

7.33 Тип BioAPI_GUI_PROGRESS

Данный тип представляет собой значение, указывающее приложению процент выполнения (окончания) осуществляемого ПБУ действия, предназначенное для отображения.

```
typedef uint8_t BioAPI_GUI_PROGRESS;
```

Примечание — Данная структура используется вместе с опцией графического интерфейса пользователя, управляемой приложением. Описание BioAPI_GUI_STATE_CALLBACK приведено в 7.36.

7.34 Тип BioAPI_GUI_RESPONSE

Данный тип возвращает значение от биометрического приложения в процессе выполнения обратного вызова.

```
typedef uint8_t BioAPI_GUI_RESPONSE;
  #define BioAPI_CAPTURE_SAMPLE (1) /* инструкция для ПБУ для получения образца */
  #define BioAPI_CANCEL (2) /* отмена операции пользователем */
  #define BioAPI_CONTINUE (3) /* выбор пользователя или приложения продолжить процесс*/
  #define BioAPI_VALID_SAMPLE (4) /*получен достоверный образец */
  #define BioAPI_INVALID_SAMPLE (5) /* получен недостоверный образец */
```

Примечание — Данная структура используется вместе с опцией графического интерфейса пользователя, управляемой приложением. Описание BioAPI_GUI_STATE_CALLBACK приведено в 7.36.

7.35 Тип BioAPI_GUI_STATE

Данный тип представляет собой битовое поле, указывающее состояние ГИП, а также то, какие другие значения параметров представлены в структуре BioAPI_GUI_STATE_CALLBACK (данную информацию ПБУ предоставляет приложению с помощью BioAPI_GUI_STATE_CALLBACK).

```
typedef uint32_t BioAPI_GUI_STATE;
  #define BioAPI_SAMPLE_AVAILABLE (0x0001) /* образец зарегистрирован и доступен */
  #define BioAPI_MESSAGE_PROVIDED (0x0002) /* сообщение для отображения, выданное ПБУ */
  #define BioAPI_PROGRESS_PROVIDED (0x0004) /* состояние процесса для отображения, выданное ПБУ */
```

Примечание — Данная структура используется вместе с опцией графического интерфейса пользователя, управляемой приложением. Описание BioAPI_GUI_STATE_CALLBACK приведено в 7.36.

7.36 Тип BioAPI_GUI_STATE_CALLBACK

7.36.1 Данный тип представляет собой указатель на функцию обратного вызова, которую предоставляет приложение для того, чтобы дать возможность ПБУ передавать через инфраструктуру приложению информацию о состоянии графического интерфейса пользователя (ГИП) и получать ответы.

7.36.2 Функция

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STATE_CALLBACK)
        (void *GuiStateCallbackCtx,
        BioAPI_GUI_STATE GuiState,
        BioAPI_GUI_RESPONSE *Response,
        BioAPI_GUI_MESSAGE Message,
        BioAPI_GUI_PROGRESS Progress,
        const BioAPI_GUI_BITMAP *SampleBuffer);
```

Возврат значения, отличающегося от BioAPI_OK (например, *BioAPI_Enroll*), приводит к немедленному возврату вызванной функции к вызывающему коду с передачей ему данного значения в качестве кода ошибки.

7.36.3 Параметры

GuiStateCallbackCtx (входной) — указатель на контекстную информацию, предоставленный инициатором запроса и возвращаемый его инициатору.

GuiState (входной) — индикация текущего состояния ПБУ по отношению к ГИП, а также индикация того, какие другие параметры являются доступными.

Response (выходной) — указатель на ответ приложения поставщику биометрической услуги при возврате от обратного вызова.

Message (входной/необязательный) — номер сообщения, подлежащего отображению пользователю. Номера и тексты сообщений не стандартизированы и зависят от ПБУ. *GuiState* указывает наличие параметра Message; если сообщения не могут быть переданы, параметр имеет нулевое значение.

Progress (входной/необязательный) — значение, которое указывает степень выполнения (в процентах) операции по созданию образца или ЗБИ. Значение может использоваться для отображения индикатора выполнения. Не все ПБУ поддерживают индикацию выполнения. *GuiState* указывает наличие значения параметра *Progress* в вызове; если степень выполнения не передается, параметр имеет нулевое значение.

SampleBuffer (входной/необязательный) — текущий буфер образца для приложения, подлежащий отображению. *GuiState* указывает наличие параметра *SampleBuffer;* если он не передается, параметр имеет нулевое значение.

Примечание — Описание интерфейса пользователя приведено в приложении С, раздел С.7.

7.37 Тип BioAPI_GUI_STREAMING_CALLBACK

7.37.1 Данный тип представляет собой указатель на функцию обратного вызова, которую предоставляет приложение для того, чтобы дать возможность ПБУ передавать поток данных для отображения в виде последовательности битовых изображений через инфраструктуру приложению.

7.37.2 Функция

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STREAMING_CALLBACK)
        (void *GuiStreamingCallbackCtx,
        const BioAPI_GUI_BITMAP *Bitmap);
```

Возврат значения, отличающегося от BioAPI_OK (например, *BioAPI_Enroll*), приведет к немедленному возврату вызванной функции к вызывающему коду с передачей ему данного значения в качестве кода ошибки.

7.37.3 Параметры

GuiStreamingCallbackCtx (входной) — указатель на контекстную информацию, предоставленный инициатором запроса и возвращаемый инициатору.

Bitmap (входной) — указатель на битовое изображение, которое должно быть отображено.

Примечание — Описание интерфейса пользователя приведено в приложении С, раздел С.7.

7.38 Тип BioAPI_HANDLE

Данный тип представляет собой уникальный идентификатор, возвращаемый в ответ на вызов функции ***BioAPI_BSPAttach,*** который идентифицирует присоединенную сессию ПБУ БиоАПИ.

```
typedef uint32_t BioAPI_HANDLE;
```

7.39 Тип BioAPI_IDENTIFY_POPULATION

Структура, используемая для идентификации набора ЗБИ, используется в качестве входного параметра функции *BioAPI_Identify* или *BioAPI_IdentifyMatch*.

```
typedef struct bioapi_identify_population {
        BioAPI_IDENTIFY_POPULATION_TYPE Type;
        union {
        BioAPI_DB_HANDLE *BIRDataBase;
        BioAPI_BIR_ARRAY_POPULATION *BIRArray;
        } BIRs;
        } BioAPI_IDENTIFY_POPULATION;
```

Если BioAPI_PRESET_ARRAY_TYPE определен в параметре *Type*, то параметр *BIRArray* должен быть выбран и установлен NULL.

7.40 Тип BioAPI_IDENTIFY_POPULATION_TYPE

Данная структура определяет метод передачи ЗБИ функции *BioAPI_Identify* или *BioAPI_IdentifyMatch* в виде массива или с помощью указателя на базу данных.

```
typedef uint8_t BioAPI_IDENTIFY_POPULATION_TYPE;
        #define BioAPI_DB_TYPE              (1)
        #define BioAPI_ARRAY_TYPE           (2)
        #define BioAPI_PRESET_ARRAY_TYPE    (3)
```

7.41 Тип BioAPI_INDICATOR_STATUS

Данная структура используется биометрическим приложением для получения и установки индикаторов устройств (модулей БиоАПИ), поддерживаемых ПБУ. Конкретная физическая форма этих индикаторов (и их наличие) зависит от исполнения.

```
typedef uint8_t BioAPI_INDICATOR_STATUS;
        #define BioAPI_INDICATOR_ACCEPT     (1)
        #define BioAPI_INDICATOR_REJECT     (2)
        #define BioAPI_INDICATOR_READY      (3)
        #define BioAPI_INDICATOR_BUSY       (4)
        #define BioAPI_INDICATOR_FAILURE    (5)
```

7.42 Тип BioAPI_INPUT_BIR

Данная структура предназначена для ввода ЗБИ БиоАПИ в ПИП, который может быть выполнен тремя способами путем передачи:

a) дескриптора ЗБИ;

b) ключа к ЗБИ в базе данных, управляемой ПБУ. Если DbHandle имеет нулевое значение, используется база данных, выбранная ПБУ. (Значение DbHandle возвращается, когда база данных ЗБИ открыта);

c) структуры ЗБИ БиоАПИ.

```
typedef struct bioapi_input_bir {
        BioAPI_INPUT_BIR_FORM Form;
        union {
            BioAPI_DBBIR_ID *BIRinDb;
            BioAPI_BIR_HANDLE *BIRinBSP;
            BioAPI_BIR *BIR;
            } InputBIR;
        } BioAPI_INPUT_BIR;
```

7.43 Тип BioAPI_INPUT_BIR_FORM

Данный тип представляет собой значение, указывающее форму или метод, с помощью которых передается ЗБИ.

```
typedef uint8_t BioAPI_INPUT_BIR_FORM;
        #define BioAPI_DATABASE_ID_INPUT        (1)
        #define BioAPI_BIR_HANDLE_INPUT         (2)
        #define BioAPI_FULLBIR_INPUT            (3)
```

Примечание — Данный тип используется как элемент BioAPI_INPUT_BIR.

7.44 Тип BioAPI_INSTALL_ACTION

Данный тип определяет действие, которое будет совершено во время операций установки ПБУ.

```
typedef uint32_t BioAPI_INSTALL_ACTION;
        #define BioAPI_INSTALL_ACTION_INSTALL       (1)
        #define BioAPI_INSTALL_ACTION_REFRESH       (2)
        #define BioAPI_INSTALL_ACTION_UNINSTALL     (3)
```

7.45 Тип BioAPI_INSTALL_ERROR

Данный тип содержит дополнительную информацию об ошибке, которая произошла во время операции установки.

```
typedef struct install_error{
        BioAPI_RETURN ErrorCode;
        BioAPI_STRING ErrorString;
} BioAPI_INSTALL_ERROR;
```

7.46 Тип BioAPI_OPERATIONS_MASK

Данный тип представляет собой битовую маску, указывающую, какие операции поддерживаются ПБУ.

```
typedef uint32_t BioAPI_OPERATIONS_MASK;
  #define BioAPI_ENABLEEVENTS                   (0x00000001)
  #define BioAPI_SETGUICALLBACKS                (0x00000002)
  #define BioAPI_CAPTURE                        (0x00000004)
  #define BioAPI_CREATETEMPLATE                 (0x00000008)
  #define BioAPI_PROCESS                        (0x00000010)
  #define BioAPI_PROCESSWITHAUXBIR              (0x00000020)
  #define BioAPI_VERIFYMATCH                    (0x00000040)
  #define BioAPI_IDENTIFYMATCH                  (0x00000080)
  #define BioAPI_ENROLL                         (0x00000100)
  #define BioAPI_VERIFY                         (0x00000200)
  #define BioAPI_IDENTIFY                       (0x00000400)
  #define BioAPI_IMPORT                         (0x00000800)
  #define BioAPI_PRESETIDENTIFYPOPULATION       (0x00001000)
  #define BioAPI_DATABASEOPERATIONS             (0x00002000)
  #define BioAPI_SETPOWERMODE                   (0x00004000)
  #define BioAPI_SETINDICATORSTATUS             (0x00008000)
  #define BioAPI_GETINDICATORSTATUS             (0x00010000)
  #define BioAPI_CALIBRATESENSOR                (0x00020000)
  #define BioAPI_UTILITIES                      (0x00040000)
  #define BioAPI_QUERYUNITS                     (0x00100000)
  #define BioAPI_QUERYBFPS                      (0x00200000)
  #define BioAPI_CONTROLUNIT                    (0x00400000)
```

Примечание — Данный тип используется как элемент BioAPI_BSP_SCHEMA.

7.47 Тип BioAPI_OPTIONS_MASK

Данный тип представляет собой битовую маску, указывающую, какие опции поддерживаются ПБУ. Необязательные функции определяются в BioAPI_OPERATIONS_MASK и в этом разделе не приводятся.

typedef uint32_t BioAPI_OPTIONS_MASK;

#define BioAPI_RAW (0x00000001)

Если установлено, то ПБУ поддерживает возврат исходных/контрольных данных.

#define BioAPI_QUALITY_RAW (0x00000002)

Если установлено, то ПБУ поддерживает возврат значения качества (в заголовке ЗБИ) для исходных биометрических данных.

#define BioAPI_QUALITY_INTERMEDIATE (0x00000004)

Если установлено, то ПБУ поддерживает возврат значения качества (в заголовке ЗБИ) для промежуточных биометрических данных.

#define BioAPI_QUALITY_PROCESSED (0x00000008)

Если установлено, то ПБУ поддерживает возврат значения качества (в заголовке ЗБИ) для обработанных биометрических данных.

#define BioAPI_APP_GUI (0x00000010)

Если установлено, то ПБУ поддерживает управление ГИП приложением.

#define BioAPI_STREAMINGDATA (0x00000020)

Если установлено, то ПБУ предоставляет потоковые данные ГИП.

#define BioAPI_SOURCEPRESENT (0x00000040)

Если установлено, то ПБУ поддерживает обнаружение присутствия биометрической характеристики в биометрическом датчике.

#define BioAPI_PAYLOAD (0x00000080)

Если установлено, то ПБУ поддерживает перенос полезной информации (сохраняет полезную информацию при выполнении функций *BioAPI_Enroll* или *BioAPI_CreateTemplate* и возвращает ее при успешном выполнении функций *BioAPI_Verify* или *BioAPI_VerifyMatch*).

#define BioAPI_BIR_SIGN (0x00000100)

Если установлено, то ПБУ возвращает ЗБИ с подписью.

#define BioAPI_BIR_ENCRYPT (0x00000200)

Если установлено, то ПБУ возвращает зашифрованные ЗБИ (часть ББД).

#define BioAPI_TEMPLATEUPDATE (0x00000400)

Если установлено, то ПБУ обновляет и предоставляет контрольный шаблон как часть операций регистрации или создания шаблона.

#define BioAPI_ADAPTATION (0x00000800)

Если установлено, то ПБУ поддерживает адаптацию ЗБИ в возвращаемых параметрах операций *BioAPI_Verify* или *BioAPI_VerifyMatch*.

#define BioAPI_BINNING (0x00001000)

Если установлено, то ПБУ поддерживает категоризацию (в операциях *BioAPI_Identify* и *BioAPI_IdentifyMatch*).

#define BioAPI_SELFCONTAINEDDEVICE (0x00002000)

Если установлено, то ПБУ поддерживает автономное устройство.

#define BioAPI_MOC (0x0004000)

Если установлено, то ПБУ напрямую поддерживает сопоставление на смарт-карте.

#define BioAPI_SUBTYPE_TO_CAPTURE (0x00008000)

Если установлено, то ПБУ поддерживает спецификацию приложения с подтипом получения данных и будет действовать в соответствии с ней.

#define BioAPI_SENSORBFP (0x00010000)

Если установлено, то ПБУ поддерживает управление модулем датчика с помощью ПБФ.

#define BioAPI_ARCHIVEBFP (0x00020000)

Если установлено, то ПБУ поддерживает управление модулем архива с помощью ПБФ.

#define BioAPI_MATCHINGBFP (0x00040000)

Если установлено, то ПБУ поддерживает управление модулем алгоритмов сопоставления с помощью ПБФ.

#define BioAPI_PROCESSINGBFP (0x00080000)

Если установлено, то ПБУ поддерживает управление модулем алгоритмов обработки с помощью ПБФ.

#define BioAPI_COARSESCORES (0x00100000)

Если установлено, то ПБУ возвращает грубо квантованную оценку (см. приложение С, пункт С.4.6).

Примечание — Данный тип используется как элемент BioAPI_BSP_SCHEMA.

7.48 Тип BioAPI_POWER_MODE

Данный тип представляет собой перечисление, определяющее типы режимов питания, которые может использовать ПБУ и присоединенные к нему устройства (модули БиоАПИ).

```
typedef uint32_t BioAPI_POWER_MODE;
        #define BioAPI_POWER_NORMAL (1) /* Доступны все функции*/
        #define BioAPI_POWER_DETECT (2) /* Способность обнаружить тип события (например —
    ввод пальца) */
        #define BioAPI_POWER_SLEEP  (3) /* Минимальный режим. Все функции выключены */
```

7.49 Тип BioAPI_QUALITY

7.49.1 Данный тип представляет собой значение, указывающее относительное качество биометрических данных в ББД.

```
typedef int8_t BioAPI_QUALITY;
```

7.49.2 Результаты биометрического анализа зависят от качества биометрического образца. Так как принятого универсального определения качества не существует, БиоАПИ определяет следующую структуру для относительной количественной оценки эффекта влияния качества на использование ЗБИ (как предполагается средством реализации ПБУ). Оценки, выдаваемые ПБУ, основаны на назначении (BioAPI_BIR_PURPOSE), определенном приложением (например, BioAPI_PURPOSE_ENROLL, BioAPI_PURPOSE_VERIFY, BioAPI_PURPOSE_IDENTIFY и т. д.). Кроме того, результаты биометрического анализа зависят от используемых приложений, то есть некоторые приложения могут требовать более высокого качества образцов, в то время как другие приложения устанавливают более низкие требования.

7.49.3 Результаты измерения качества возвращаются как целые значения в диапазоне 0—100, за исключением следующих случаев:

Значение «минус 1»: BioAPI_QUALITY не был установлен ПБУ (для объяснения следует обратиться к документации разработчика ПБУ).

Значение «минус 2»: BioAPI_QUALITY не поддерживается ПБУ.

7.49.4 Обратную связь BioAPI_QUALITY и биометрического приложения используют с целью:

c)* Основная цель — ПБУ должен сообщить приложению, насколько подходит биометрическая выборка для назначения (BioAPI_PURPOSE), указанного приложением (как предполагается разработчиком ПБУ на основе использования сценария, предположенного разработчиком этого ПБУ).

d) Вторичная цель — предоставить приложению относительные результаты (например, текущий образец лучше или хуже предыдущего образца).

7.49.5 Значения показателя качества в диапазоне 0 — 100 имеют следующую общую интерпретацию:

0 — 25 НЕДОПУСТИМОЕ — ББД не может использоваться для назначения, указанного приложением (BioAPI_PURPOSE). ББД должен быть заменен с использованием одного или нескольких новых биометрических образцов;

26 — 50 НЕУДОВЛЕТВОРИТЕЛЬНОЕ — ББД обеспечит плохое выполнение задачи, указанной приложением (BioAPI_PURPOSE), и в большинстве случаев поставит под угрозу задачу приложения. ББД должен быть заменен с использованием одного или нескольких новых биометрических образцов;

51 — 75 УДОВЛЕТВОРИТЕЛЬНОЕ — ББД обеспечит хорошее выполнение задачи в большинстве прикладных программ, основанных на назначении, указанном приложением (BioAPI_PURPOSE). Приложение должно пытаться получить более качественные данные, если разработчик приложения предусматривает более требовательное использование;

76 — 100 ПРЕВОСХОДНОЕ — ББД обеспечит хорошее выполнение задачи, указанной приложением (BioAPI_BIR_PURPOSE).

Примечание — BioAPI_QUALITY соответствует «BDB_Quality» (качество ББД) в ЕСФОБД по ИСО/МЭК 19785-1.

---

* *В оригинале ИСО/МЭК 19784-1 допущена ошибка. Нумерация перечислений начинается с пункта c).*

7.50 Тип BioAPI_RETURN

7.50.1 Данный тип содержит данные, возвращаемые всеми функциями БиоАПИ. Допустимые значения включают в себя:

- BioAPI_OK;
- все ошибочные значения, определенные в настоящем стандарте;
- особые значения ошибок ПБУ, определенные и возвращенные ПБУ;
- все значения ошибок, определенные для компонентов более низких уровней;
- зависящие от модулей БиоАПИ значения ошибок, определенные и возвращенные ПБУ.

```
typedef uint32_t BioAPI_RETURN;
        #define BioAPI_OK                    (0)
```

7.50.2 Определения

BioAPI_OK — указывает, что операция была выполнена успешно.

Любое другое значение — указывает, что операция не была выполнена успешно и идентифицирует обнаруженную ошибку, которая привела к аварийному завершению (перечень стандартизированных кодов ошибок указан разделе 11).

7.51 Тип BioAPI_STRING

7.51.1 Данный тип используется структурами данных БиоАПИ для представления символьной строки в буфере фиксированной длины. Символьная строка должна завершаться нулевым символом.

```
typedef uint8_t BioAPI_STRING [269];
```

7.51.2 Данная строка должна состоять из символов, описанных в ИСО/МЭК 10646, кодировки UTF-8 по ИСО/МЭК 10646, приложение D.

7.52 Тип BioAPI_TIME

Данный тип определяет время создания ЗБИ.

```
typedef struct bioapi_time {
        uint8_t Hour;     /* диапазон допустимых значений: 00 — 23, 99 */
        uint8_t Minute;   /* диапазон допустимых значений: 00 — 59, 99 */
        uint8_t Second;   /* диапазон допустимых значений: 00 — 59, 99 */
} BioAPI_TIME;
  #define BioAPI_NO_HOUR_AVAILABLE          (99)
  #define BioAPI_NO_MINUTE_AVAILABLE        (99)
  #define BioAPI_NO_SECOND_AVAILABLE        (99)
```

Условие NO VALUE AVAILABLE (данные недоступны) должно быть указано установкой всех полей на «99» (десятичные). При записи данных года, месяца, дня в заголовке ЗБИ и недоступной информации о времени должны быть использованы значения BioAPI_NO_HOUR_AVAILABLE, BioAPI_NO_MINUTE_AVAILABLE и BioAPI_NO_SECOND_AVAILABLE.

Примечание — Формат времени соответствует установленному в ISO 8601 [2].

7.53 Тип BioAPI_UNIT_ID

Данный тип представляет собой ИД модуля БиоАПИ — 32-разрядное целое число, присвоенное модулю БиоАПИ от ПБУ, который управляет (непосредственно или косвенно) модулем БиоАПИ.

```
typedef uint32_t BioAPI_UNIT_ID;
  #define BioAPI_DONT_CARE                  (0x00000000)
  #define BioAPI_DONT_INCLUDE               (0xFFFFFFFF)
```

7.54 Тип BioAPI_UNIT_LIST_ELEMENT

7.54.1 Данный тип указывает модуль БиоАПИ по категории ИД. Список данных элементов используется для установления присоединенной сессии.

```
typedef struct bioapi_unit_list_element {
        BioAPI_CATEGORY UnitCategory;
        BioAPI_UNIT_ID UnitId;
} BioAPI_UNIT_LIST_ELEMENT;
```

7.54.2 Определения

UnitCategory — определяет категорию модуля БиоАПИ.

UnitId — ИД модуля БиоАПИ, используемый в данной присоединенной сессии, создаваемый ПБУ и являющийся уникальным.

7.55 Тип BioAPI_UNIT_SCHEMA

7.55.1 Данный тип представляет собой схему модуля БиоАПИ и показывает специфические характеристики модуля БиоАПИ.

```
typedef struct bioapi_unit_schema {
        BioAPI_UUID BSPUuid;
        BioAPI_UUID UnitManagerUuid;
        BioAPI_UNIT_ID UnitId;
        BioAPI_CATEGORY UnitCategory;
        BioAPI_UUID UnitProperties;
        BioAPI_STRING VendorInformation;
        uint32_t SupportedEvents;
        BioAPI_UUID UnitPropertyID;
        BioAPI_DATA UnitProperty;
        BioAPI_STRING HardwareVersion;
        BioAPI_STRING FirmwareVersion;
        BioAPI_STRING SoftwareVersion;
        BioAPI_STRING HardwareSerialNumber;
        BioAPI_BOOL AuthenticatedHardware;
        uint32_t MaxBSPDbSize;
        uint32_t MaxIdentify;
} BioAPI_UNIT_SCHEMA;
```

Примечание — Схема модуля БиоАПИ используется как параметр функции ***BioAPI_QueryUnits*** и ***BioAPI_EventHandler*** но не хранится в реестре компонентов.

7.55.2 Определения

BspUuid — УУИД ПБУ, поддерживающего этот модуль БиоАПИ.

UnitManagerUuid — УУИД программного компонента (либо ПБУ, либо ПБФ), непосредственно управляющего модулем БиоАПИ.

UnitId — ИД модуля БиоАПИ в данной присоединенной сессии, создаваемый ПБУ и являющийся уникальным.

UnitCategory — определяет категорию модуля БиоАПИ.

UnitProperties — УУИД, указывающий набор свойств модуля БиоАПИ. Указанный набор может быть определен изготовителем или соответствовать конкретному стандарту.

VendorInformation — содержит собственную информацию изготовителя.

SupportedEvents — маска, указывающая, какие типы событий поддерживаются аппаратными средствами.

UnitPropertyID — УУИД формата следующей структуры свойств модуля.

UnitProperty — адрес и длина буфера памяти, содержащего свойства модуля, описывающие модуль БиоАПИ. Формат и содержание свойств модуля могут быть либо описаны изготовителем, либо указаны в связанном стандарте.

HardwareVersion — строка, оканчивающаяся пустым символом, содержащая версию аппаратного обеспечения; может быть пустой, если параметр недоступен.

FirmwareVersion — строка, оканчивающаяся пустым символом, содержащая версию встроенного программного обеспечения; может быть пустой, если параметр недоступен.

SoftwareVersion — строка с нулевым символом на конце, содержащая версию программного обеспечения; может быть пустой, если параметр недоступен.

HardwareSerialNumber — строка с нулевым символом на конце, содержащая уникальный серийный номер, определяющий изготовителя компонентов аппаратного обеспечения; может быть пустой, если параметр недоступен.

AuthenticatedHardware — Булево значение, указывающее, был ли аутентифицирован компонент аппаратного обеспечения.

MaxBSPDbSize — максимальный размер поддерживаемой модулем БиоАПИ базы данных; если равен нулю — база данных не существует.

MaxIdentify — максимальная совокупность идентификации, поддерживаемая модулем БиоАПИ; безразмерная — «FFFFFFFF».

7.56 Тип BioAPI_UUID

Данный тип представляет собой универсальный уникальный идентификатор, используемый для идентификации и обращения к компонентам (ПБУ, ПБФ, модулям и инфраструктуре БиоАПИ), идентификации базы данных ЗБИ, а также и в качестве индекса базы данных ЗБИ, управляемой ПБУ .

typedef uint8_t BioAPI_UUID[16];

Примечание 1 — УУИД определены в ИСО/МЭК 9834-8.

Примечание 2 — УУИД БиоАПИ заголовка ЗБИ соответствует параметру «CBEFF_BDB_index» по ИСО/МЭК 19785-1.

7.57 Тип BioAPI_VERSION

7.57.1 Данный тип используется для представления версии спецификации БиоАПИ или определения того, с какими компонентами или данными совместимы ПБФ. Данное значение обычно используется в заголовке ЗБИ и схемах реестра компонентов.

Данная версия БиоАПИ имеет целое значение (десятичное) 32 или (шестнадцатиричное) 20, соответствующее основному значению 2 и дополнительному — 0.

typedef uint8_t BioAPI_VERSION;

Примечание 1 — Данный тип не используется для версии продукта, который обычно представлен в виде строки.

Примечание 2 — Версия БиоАПИ используется в заголовке ЗБИ и соответствует параметру «CBEFF_patron_header_version» ИСО/МЭК 19785-1.

7.57.2* Определение

Версия БиоАПИ объединяет номер редакции и номер поправки или изменения данной редакции: первая шестнадцатеричная цифра определяет номер редакции, вторая — номер поправки или изменения данной редакции:

BioAPI_VERSION 0xnm,

где n — номер редакции;

m — номер поправки или изменения данной редакции.

8 Функции БиоАПИ

8.1 Функции управления компонентом

Функции, указанные в этом разделе, обрабатываются инфраструктурой БиоАПИ. Некоторые из них обрабатываются внутренне, а другие передаются ПБУ через ИПУ.

8.1.1 Функция BioAPI_Init

BioAPI_RETURN BioAPI BioAPI_Init
(BioAPI_VERSION Version);

8.1.1.1 Описание

Данная функция инициализирует инфраструктуру БиоАПИ и проверяет, совместима ли версия инфраструктуры БиоАПИ, ожидаемой приложением, с версией инфраструктуры БиоАПИ в системе. Данная функция должна быть вызвана приложением, по крайней мере, один раз.

Любой вызов функции ***BioAPI_Init,*** когда предыдущий вызов функции ***BioAPI_Init*** не был закончен связанным вызовом функции ***BioAPI_Terminate,*** будет обработан следующим образом: инфраструктура БиоАПИ выдаст ответ BioAPI_OK (только в том случае, если номер версии инфраструктуры совместим с номером версии инфраструктуры, которая была предложена предыдущим вызовом функции ***BioAPI_Init***) или BioAPIERR_INCOMPATIBLE_VERSION, но не будет переинициализирована. Счетчик числа удачных вызовов ***BioAPI_Init*** будет сохраняться инфраструктурой и не завершится до тех пор, пока число соответствующих вызовов *BioAPI_Terminate* не достигнет определенного значения.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБУ.

8.1.1.2 Параметры

Version (входной) — номер редакции и номер поправки спецификации БиоАПИ, с которой совместимо биометрическое приложение.

** В оригинале ИСО/МЭК 19784-1 допущена ошибка, должен быть номер 7.57.2.*

8.1.1.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки в противном случае. Значение BioAPI_OK указывает на отсутствие ошибки и успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.1.1.4 Ошибки

BioAPIERR_INCOMPATIBLE_VERSION

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.2 Функция BioAPI_Terminate

BioAPI_RETURN BioAPI BioAPI_Terminate (void);

8.1.2.1 Описание

Данная функция завершает использование инфраструктуры БиоАПИ. Инфраструктура может произвести очистку внутреннего состояния, связанного с запрашивающим приложением.

Данная функция может быть вызвана только в том случае, если функция ***BioAPI_Init,*** для которой еще не была вызвана рассматриваемая функция, была ранее удачно вызвана, по крайней мере, один раз. Функция ***BioAPI_Terminate*** не должна вызываться до успешного выполнения функции ***BioAPI_Init.***

Данная функция не должна вызываться приложением, если был сделан вызов функции ***BioAPI_BSPLoad,*** для которого не был выполнен соответствующий вызов функции ***BioAPI_BSPUnload*** (для данного УУИД ПБУ). Если данная функция должна быть вызвана во время существования загруженных ПБУ, то для каждого вызова функций ***BioAPI_BSPLoad,*** для которого не был выполнен вызов функции ***BioAPI_BSPUnload,*** инфраструктура БиоАПИ должна неявно выполнить действия, соответствующие отсутствующему вызову ***BioAPI_BSPUnload*** (как если бы соответствующая функция была вызвана), а затем выполнить действия, соответствующие ***BioAPI_Terminate*** (т. е. инфраструктура должна выгрузить все ПБУ до завершения использования инфраструктуры).

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБУ, за исключением тех случаев, когда подразумевается выполнение функции ***BioAPI_BSPUnload,*** как указано выше.

8.1.2.2 Параметры

Отсутствуют.

8.1.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

Примечание — Обычно ожидается успешное выполнение функции ***BioAPI_Terminate;*** однако при возникновении непредвиденных условий приложение может не предпринимать дальнейшие действия.

8.1.2.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.3 Функция BioAPI_GetFrameworkInfo

BioAPI_RETURN BioAPI BioAPI_GetFrameworkInfo
(BioAPI_FRAMEWORK_SCHEMA *FrameworkSchema);

8.1.3.1 Описание

Данная функция возвращает информацию об инфраструктуре БиоАПИ. Так как на компьютере могут быть установлены несколько инфраструктур, приложению требуется информация о них, чтобы выбрать одну для использования.

8.1.3.2 Параметры

FwUuid (выходной) — УУИД инфраструктуры, который должен быть одинаковым для всех экземпляров данной инфраструктуры.

FrameworkSchema (выходной) — указатель на область памяти, в которую будет возвращена информация о схеме.

8.1.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.3.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.4 Функция BioAPI_EnumBSPs

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

8.1.4.1 Описание

Данная функция предоставляет информацию о всех ПБУ, установленных в реестре компонентов. Функция выполняет действия в следующем порядке:

a) выделяет блок памяти, достаточный для хранения массива элементов типа BioAPI_BSP_SCHEMA с числом элементов, равным числу установленных ПБУ;

b) заполняет массив схемами установленных ПБУ;

c) возвращает адрес массива в параметре BSPSchemaArray и число элементов массива в параметре NumberOfElements.

Данная функция может быть вызвана только в том случае, если был сделан, по крайней мере, один вызов функции ***BioAPI_Init,*** для которого еще не был сделан соответствующий вызов функции ***BioAPI_Terminate.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

Блок памяти, содержащий массив, должен быть освобожден приложением путем вызова функции ***BioAPI_Free*** (8.7.2), если приложение в дальнейшем не будет его использовать. Те элементы массива, на которые указывает блок памяти путем параметра *Path*, также должны быть освобождены приложением с помощью вызова функции ***BioAPI_Free,*** если приложение больше не будет использовать элементы массива.

8.1.4.2 Параметры

BspSchemaArray (выходной) — указатель на адрес массива элементов типа BioAPI_BSP_SCHEMA (распределенного инфраструктурой), содержащего информацию о схеме ПБУ.

NumElementsReturned (выходной) — указатель на число элементов массива (равно числу схем ПБУ в реестре компонентов).

8.1.4.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.4.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.5 Функция BioAPI_BSPLoad

```
BioAPI_RETURN BioAPI BioAPI_BSPLoad
  (const BioAPI_UUID *BSPUuid,
  BioAPI_EventHandler AppNotifyCallback,
  void* AppNotifyCallbackCtx);
```

8.1.5.1 Описание

Данная функция инициализирует ПБУ с помощью функции ***BioSPI_BSPLoad*** (см. 9.3.1.1). Инициализация включает в себя регистрацию обработчика события приложения для указанного ПБУ и разрешение всех событий. Приложение может предоставить функцию обработчика событий, чтобы получать уведомления о событиях. Несколько приложений могут независимо и одновременно загружать один и тот же ПБУ, и каждое приложение может установить свой собственный обработчик событий. Все приложения должны получать уведомление о событии. Если приложение загружает несколько ПБУ, то могут использоваться одни и те же или разные обработчики события.

Приложение может установить столько обработчиков событий, сколько необходимо для данного ПБУ, путем многократного вызова функции ***BioAPI_BSPLoad*** для данного ПБУ. Обработчик событий идентифицируется по его адресу и содержанию.

Если в ПБУ возникает событие, то ПБУ может послать уведомление о событии инфраструктуре путем вызова обработчика событий инфраструктуры.

Когда инфраструктура получает уведомление о событии от ПБУ, она должна послать уведомление каждому обработчику событий, установленному каждым приложением, для которого данное уведомление о событии возможно для данного ПБУ. Поэтому отдельный обратный вызов уведомления о событии, сделанный из ПБУ к инфраструктуре, может привести к появлению нулевого значения или к обратным вызовам инфраструктуры к приложениям.

Когда инфраструктура получает уведомление о событии от ПБУ, она должна вызвать все обработчики событий, установленные каждым приложением для данного ПБУ. Если приложение установило несколько обработчиков событий, они должны быть вызваны по одному в любом порядке, выбранном инфраструктурой (такой вызов предпочтительнее одновременного вызова).

Уведомление о событии может появиться в любое время: во время вызовов БиоАПИ или в то время, когда вызовы БиоАПИ не выполняются. Разработчик приложения должен гарантировать, что обратные вызовы будут правильно и безопасно обрабатываться приложением независимо от того, когда приложение будет их получать.

Примечание — В этом случае необходимо использовать технологию синхронизированных потоков и порядка действий, сформированных кодом приложения на обработчик события.

Применяют «счетчик использований» установки обработчиков событий; они должны быть удалены путем вызова функции ***BioAPI_BSPUnload*** столько раз, сколько были установлены. Если ПБУ загружен путем вызова функции ***BioAPI_BSPLoad,*** он должен для каждого существующего модуля БиоАПИ немедленно вызвать событие «установка». Это событие укажет биометрическому приложению, что оно может продолжать работу и вызвать функцию ***BioAPI_BSPAttach.*** Если аппаратный компонент для определенных функциональных возможностей отсутствует, то событие «установка» не может быть инициировано до тех пор, пока аппаратный компонент не будет подключен.

Данная функция может быть вызвана только в том случае, если был выполнен, по крайней мере, один вызов функции ***BioAPI_Init,*** для которого не был выполнен соответствующий вызов функции ***BioAPI_Terminate.*** Функция ***BioAPI_BSPAttach*** может быть вызвана несколько раз после выполнения функции ***BioAPI_BSPLoad.***

Функция ***BioAPI_BSPLoad*** не должна вызываться до тех пор, пока ПБУ не будет установлен с помощью функции ***BioAPI_Util_InstallBSP.*** Определение установки ПБУ может быть проведено вызовом функции ***BioAPI_EnumBSPs.***

8.1.5.2 Параметры

BSPUuid (входной) — УУИД ПБУ, выбранного для загрузки.

AppNotifyCallback (входной / необязательный) — функция уведомления о событии, поставляемая вызывающим кодом. Определяет обратный вызов для уведомлений о событиях от загруженного (и позже присоединенного) ПБУ.

AppNotifyCallbackCtx (входной) — указатель на контекстную информацию. Когда выбранный поставщик услуги инициирует событие, данное значение передается как входной параметр обработчику события, указанному в *AppNotifyCallback*.

8.1.5.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.5.4 Ошибки

BioAPIERR_BSP_LOAD_FAIL

BioAPIERR_INVALID_UUID

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.6 Функция BioAPI_BSPUnload

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

8.1.6.1 Описание

Данная функция выполняет отмену регистрации функции обратного вызова уведомлений о событиях вызывающего кода, указанного в BSPUuid. Функция ***BioAPI_BSPUnload*** аналогична вызову функции ***BioAPI_BSPLoad.*** Если все обратные вызовы, зарегистрированные БиоАПИ, удалены, то БиоАПИ выгружает (для этого биометрического приложения) ПБУ, загруженный путем вызова функции ***BioAPI_BSPLoad.***

Для однозначного определения зарегистрированных обратных вызовов инфраструктура БиоАПИ использует три входных параметра: BSPUuid, AppNotifyCallback и AppNotifyCallbackCtx.

Данная функция может быть вызвана (для данного УУИД ПБУ) только в том случае, если был выполнен, по крайней мере, один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого не был выполнен соответствующий вызов данной функции.

Данную функцию не следует вызывать, если был выполнен вызов функции ***BioAPI_BSPAttach,*** для которого еще не был осуществлен вызов функции ***BioAPI_BSPDetach*** (для данного дескриптора ПБУ). Если данная функция должна быть вызвана, когда ПБУ все еще присоединен, то для каждого вызова функции ***BioAPI_BSPAttach,*** для которого не был выполнен вызов функции ***BioAPI_BSPDetach,*** инфраструктура БиоАПИ должна неявно выполнить действия, соответствующие отсутствующему вызову функции ***BioAPI_BSPDetach*** (как будто соответствующая функция была вызвана в это время), а затем выпол-

нить действия, соответствующие вызову функции ***BioAPI_BSPUnload*** (т. е. инфраструктура должна отсоединить ПБУ до его выгрузки).

Это же происходит и в случае, когда действия, соответствующие отсутствующему вызову функции ***BioAPI_BSPUnload,*** неявно выполнены инфраструктурой во время вызова функции ***BioAPI_Terminate*** (см. 8.1.2).

8.1.6.2 Параметры

BSPUuid (входной) — УУИД БСП, выбранного для выгрузки.

AppNotifyCallback (входной / необязательный) — функция уведомления о событии, для которой будет выполнена отмена регистрации. Функция, которая была передана при вызове функции ***BioAPI_BSPLoad.***

AppNotifyCallbackCtx (входной / необязательный) — контекст уведомления о событии, который был предоставлен в соответствующем вызове функции ***BioAPI_BSPLoad.***

8.1.6.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.6.4 Ошибки

BioAPIERR_INVALID_UUID

BioAPIERR_BSP_NOT_LOADED

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.7 Функция BioAPI_BSPAttach

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

8.1.7.1 Описание

Данная функция выполняет присоединение ПБУ и проверяет совместимость ожидаемой приложением версии ПБУ и версии системы. Вызывающий код должен определить список, состоящий из нуля или более модулей БиоАПИ, которые должен использовать вызов ПБУ в создаваемой присоединенной сессии.

Данная функция может быть вызвана (для данного УУИД ПБУ) только в том случае, если был выполнен, по крайней мере, один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого не был выполнен соответствующий вызов функции ***BioAPI_BSPUnload.*** Функция ***BioAPI_BSPAttach*** может быть вызвана несколько раз при каждом вызове функции ***BioAPI_BSPLoad*** (перед вызовом функции ***BioAPI_BSPUnload***) для одного и того же ПБУ, создавая несколько вызовов данного ПБУ.

8.1.7.2 Параметры

BSPUuid (входной) — указатель на структуру BioAPI_UUID, содержащую глобальный уникальный идентификатор (УУИД) для ПБУ.

Version (входной) — номера редакции и поправки данной спецификации БиоАПИ, которую приложение предлагает ПБУ для поддержки. ПБУ должен определить, совместимы ли его услуги с номером версии приложения.

UnitList (входной) — указатель на буфер, содержащий список структур BioAPI_UNIT_LIST_ELEMENT, указывающий ПБУ, какие модули БиоАПИ (поддерживаемые ПБУ) он должен поддерживать в данной прикрепленной сессии. Структуры содержат ИД и категорию каждого модуля БиоАПИ. Для каждой категории модулей БиоАПИ приложение может выбрать одну из следующих возможностей:

a) выбор специального модуля БиоАПИ: для определения особого модуля БиоАПИ для использования в данной прикрепленной сессии ИД и категория данного модуля БиоАПИ должна быть предоставлена этому элементу;

b) выбор любого модуля БиоАПИ: когда параметр UnitID установлен в BioAPI_DONT_CARE в специальном элементе, ПБУ выберет модуль БиоАПИ данной категории или возвратит ошибку, если он не поддерживает какой-либо модуль БиоАПИ данной категории. Если особая категория не включена в список, ПБУ также выберет для использования модуль БиоАПИ данной категории, если он ее поддерживает (однако ошибка не будет возвращена, если модуль БиоАПИ не поддерживает ее);

c) выбор отсутствия модуля БиоАПИ: когда параметр UnitID установлен в BioAPI_DONT_CARE, ПБУ не присоединит модуль БиоАПИ данной категории, даже если он ее поддерживает.

Примечание — Любой следующий вызов, требующий использования модуля БиоАПИ данной категории, закончится возвращением ошибки.

NumUnits (входной) — число элементов модулей БиоАПИ в списке, на который указывает указатель UnitList. Если этот параметр содержит нулевое значение, то ПБУ выберет модуль БиоАПИ любой категории, которыми ПБУ управляет напрямую или косвенно.

Примечание — При каждом вызове биометрическим приложением ПБУ может быть присоединен только один модуль БиоАПИ любой категории.

NewBSPHandle (выходной) — новый дескриптор, который может использоваться для взаимодействия с запрошенным ПБУ. Если выполнение функции заканчивается ошибкой, будет установлено значение BioAPIERR_FRAMEWORK_INVALID_BSP_HANDLE.

8.1.7.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет детальное условие ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают условия ошибки.

8.1.7.4 Ошибки

BioAPIERR_INCOMPATIBLE_VERSION
BioAPIERR_BSP_NOT_LOADED
BioAPIERR_INVALID UNIT_ID
BioAPIERR_INVALID_UUID
BioAPIERR_UNIT_IN_USE
BioAPIERR_INVALID_CATEGORY

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.8 Функция BioAPI_BSPDetach

```
BioAPI_RETURN BioAPI BioAPI_BSPDetach
(BioAPI_HANDLE BSPHandle);
```

8.1.8.1 Описание

Данная функция отсоединяет биометрическое приложение от вызова ПБУ.

При выполнении данной функции все распределенные ресурсы ПБУ, связанные с прикрепленной сессией ПБУ, должны быть освобождены или признаны недействительными. Это особенно важно для ЗБИ, дескрипторов ПБУ и дескрипторов базы данных. При выполнении данной функции все установленные функции обратного вызова будут недействительными.

Данная функция может быть вызвана только после вызова функции ***BioAPI_BSPAttach*** и должна вызываться не более одного раза для одного и того же дескриптора ПБУ, созданного вызовом функции ***BioAPI_BSPAttach.***

8.1.8.2 Параметры

BSPHandle (входной) — дескриптор, идентифицирующий присоединенную сессию ПБУ, которая должна быть прекращена.

8.1.8.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.8.4 Ошибки

BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.9 Функция BioAPI_QueryUnits

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

8.1.9.1 Описание

Данная функция предоставляет информацию обо всех модулях БиоАПИ, которые непосредственно или косвенно управляются ПБУ, идентифицированных предоставляемым УУИД ПБУ и присутствующих в системе. Функция выполняет действия в следующем порядке:

a) устанавливает число модулей БиоАПИ, которые будут управляться непосредственно или косвенно данным ПБУ;

b) выделяет блок памяти, достаточный для размещения массива элементов типа BioAPI_UNIT_SCHEMA с числом элементов, равным числу модулей БиоАПИ, определенному в перечислении a);

c) заполняет массив схемами всех модулей БиоАПИ, определенных в перечислении a);

d) возвращает адрес массива в параметре UnitSchemaArray и размер массива в параметре NumberOfElements.

Примечание — Когда инфраструктура вызывает функцию ПБУ BioSPI_QueryUnits, ПБУ выделяет память для данных, которые должны быть возвращены инфраструктуре. В некоторых реализациях архитектуры инфраструктура передает данные приложению в точности о том, как возвращено ПБУ, так как приложение интерпретирует адрес так же, как и ПБУ, и сможет получить доступ к данным, которые размещены по этому адресу. В других реализациях архитектуры инфраструктура должна переместить все данные, возвращенные ПБУ, во вновь выделенный блок памяти, доступный приложению, и вызвать функцию BioSPI_Free после копирования каждого блока памяти, но перед возвращением вызова BioAPI_QueryUnits. В первом случае, когда приложение вызывает функцию BioAPI_Free, инфраструктура делает соответствующий вызов BioSPI_Free. Во втором случае вызов BioAPI_Free обрабатывается самой инфраструктурой. Однако особенности данных реализаций инфраструктуры не видны приложению.

Блок памяти, содержащий массив, должен быть освобожден приложением путем вызова функции ***BioAPI_Free*** (см. 8.7.2), если он больше не нужен приложению. Блок памяти, на который ссылаются элементы UnitProperties в пределах каждого элемента массива, должен также быть освобожден приложением путем вызова функции ***BioAPI_Free***, если он больше не нужен приложению. Данная функция может быть вызвана только после вызова функции ***BioAPI_Load*** для ПБУ и не может быть вызвана после вызова функции ***BioAPI_Unload*** для ПБУ.

8.1.9.2 Параметры

BSPUuid (входной) — УУИД ПБУ, на который должна быть возвращена информация о модуле.

UnitSchemaArray (выходной) — указатель на массив элементов типа BioAPI_UNIT_SCHEMA (распределенный ПБУ, см. примечание к 8.1.9.1), содержащий информацию о схемах модуля.

NumberOfElements (выходной) — указатель на число элементов массива.

8.1.9.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.9.4 Ошибки

BioAPIERR_BSP_NOT_LOADED

BioAPIERR_INVALID_UUID

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.10 Функция BioAPI_EnumBFPs

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

8.1.10.1 Описание

Данная функция предоставляет информацию обо всех ПБУ, установленных в данное время в реестре компонентов, и выполняется в следующем порядке:

a) выделяет область памяти, достаточную для размещения массива элементов типа BioAPI_BFP_SCHEMA с числом элементов, равным числу установленных ПБУ;

b) заполняет массив схемами всех установленных ПБУ;

c) возвращает адрес массива в параметре BSPSchemaArray и размер массива в параметре NumberOfElements.

Данная функция может быть вызвана только в том случае, если был сделан, по крайней мере, один вызов функции ***BioAPI_Init,*** для которого еще не был сделан соответствующий вызов функции ***BioAPI_Terminate.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

Область памяти, содержащая массив, должна быть освобождена приложением с помощью вызова функции ***BioAPI_Free*** (см. 8.7.2) в том случае, если он больше не нужен приложению.

Область памяти, на которую указывают параметры Path и BFPProperty в пределах каждого элемента массива, также должна быть освобождена приложением путем вызова функции ***BioAPI_Free***, если элементы массива не используются приложением.

8.1.10.2 Параметры

BfpSchemaArray (выходной) — указатель на массив элементов типа BioAPI_UNIT_SCHEMA (распределенный инфраструктурой), содержащих информацию о схемах ПБУ.

NumberOfElements (выходной) — указатель на число элементов массива, равное числу схем ПБУ в реестре компонентов.

8.1.10.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.10.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.11 Функция BioAPI_QueryBFPs

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

8.1.11.1 Описание

Данная функция возвращает список ПБФ, которые установлены в данный момент в реестре компонентов и поддерживаются ПБУ, идентифицированным УУИД ПБУ. Функция выполняет действия в следующем порядке:

a) определяет число установленных ПБФ, поддерживаемых данным ПБУ;

b) выделяет область памяти, достаточную для размещения массива элементов типа BioAPI_BFP_LIST_ELEMENT с числом элементов ПБФ, определенным в перечислении a);

c) заполняет массив идентификационной информацией (категория и УУИД) о ПБФ, определенных в перечислении a);

d) возвращает адрес массива в параметре BFPList и число элементов массива в параметре NumberOfElements.

Примечание — Когда инфраструктура вызывает функцию ПБУ BioSPI_QueryBSPs, ПБУ выделяет память для данных, которые должны быть возвращены инфраструктуре. В некоторых реализациях архитектуры инфраструктура передает данные приложению точно так же, как их возвращает ПБУ, так как приложение интерпретирует адрес так же, как и ПБУ и сможет получить доступ к данным, которые размещены ПБУ по этому адресу. В других реализациях архитектуры инфраструктура должна переместить все данные, возвращенные ПБУ, во вновь выделенную область памяти, доступную приложению, и вызвать функцию BioSPI_Free после копирования каждого блока памяти, но перед возвращением вызова BioAPI_QueryBSP. В первом случае, когда приложение вызывает функцию BioAPI_Free, инфраструктура делает соответствующий вызов BioSPI_Free. Во втором случае вызов функции BioAPI_Free обрабатывается инфраструктурой. Однако различия реализации инфраструктуры приложение не обнаруживает.

Дополнительная информация о поддерживаемых ПБФ может быть получена путем вызова функции ***BioAPI_EnumBFPs*** и анализом параметра BFPSchemaArray по соответствующим BfpUuids.

Данная функция может быть вызвана только после вызова функции ***BioAPI_Load*** для специального ПБУ и не может быть вызвана после вызова функции ***BioAPI_Unload*** для ПБУ.

Область памяти, содержащая массив, должна быть освобождена приложением путем вызова функции ***BioAPI_Free*** (см. 8.7.2) в том случае, если приложение больше не использует элементы массива.

8.1.11.2 Параметры

BSPUuid (входной) — УУИД ПБУ, по которому должна быть возвращена информация о ПБФ.

BFPList (выходной) — указатель на массив элементов типа BioAPI_BFP_LIST_ELEMENT (распределенный ПБУ, см. примечание выше) и буфер, содержащий идентификационную информацию о ПБФ.

NumberOfElements (выходной) — указатель на число элементов массива.

8.1.11.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.11.4 Ошибки

BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.12 Функция BioAPI_ControlUnit

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

8.1.12.1 Описание

Данная функция посылает управляющие данные от приложения модулю БиоАПИ и получает обратно данные состояния или рабочие данные. Содержание параметра *ControlCode* посылаемых (входных) и получаемых (выходных) данных должно быть определено в спецификации на интерфейс для данного модуля БиоАПИ или связанного ИПФ в том случае, если он присутствует.

Данная функция выделяет область памяти, достаточную для размещения выходных данных, которые должны быть возвращены приложению, заполняет блок данными и записывает в поля Length и Data структуры OutputData размер и адрес блока памяти соответственно.

Блок памяти, возвращенный при вызове функции БиоАПИ, должен быть освобожден приложением путем вызова функции ***BioAPI_Free*** (8.7.2).

8.1.12.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

UnitId — ИД модуля БиоАПИ.

ControlCode (входной) — код функции в вызываемом модуле БиоАПИ.

InputData (входной) — указатель на структуру BioAPI-DATA, содержащую адрес и длину буфера данных, которые должны быть посланы модулю БиоАПИ в соответствии с данным *ControlCode*.

OutputData (выходной) — указатель на структуру BioAPI-DATA. На выходе она должна содержать адрес и длину буфера данных, содержащего данные, полученные от модуля БиоАПИ после обработки функции, указанной в *ControlCode*. Если функция не выделила область памяти, то адрес должен быть установлен на пустой указатель, а длина буфера задана равной нулю.

8.1.12.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.12.4 Ошибки

BioAPIERR_BIOAPI_UNIT_NOT_INSERTED
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_IN_USE
BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.2 Операции над дескриптором данных

8.2.1 Операция BioAPI_FreeBIRHandle

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

8.2.1.1 Описание

Данная функция освобождает память и ресурсы, связанные с указанным дескриптором ЗБИ. Связанная с вызовом данной функции ЗБИ больше не может быть получена по этому дескриптору. При необходимости биометрическое приложение может сохранить ЗБИ в базе данных, управляемой ПБУ, (путем вызова функции ***BioAPI_DbStoreBIR***) перед вызовом функции ***BioAPI_FreeBIRHandle.*** В качестве альтернативы приложение может вызвать функцию ***BioAPI_GetBIRFromHandle,*** которая восстановит ЗБИ и очистит дескриптор, вместо вызова функции ***BioAPI_FreeBIRHandle.***

Данная функция может быть вызвана только после вызова функции ***BioAPI_BSPAttach*** и не может быть вызвана по окончании выполнения функции ***BioAPI_BSPDetach*** для дескриптора ПБУ, созданного при выполнении функции ***BioAPI_BSPAttach.***

8.2.1.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

Handle (входной) — освобождаемый дескриптор ЗБИ.

8.2.1.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.2.1.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.2.2 Операция BioAPI_GetBIRFromHandle

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

8.2.2.1 Описание

Данная функция извлекает ЗБИ, связанную с дескриптором ЗБИ, возвращаемым ПБУ. Перед окончанием выполнения функции, ПБУ освобождает дескриптор ЗБИ.

8.2.2.2 Параметры

BSPHandle (входной) — дескриптор ПБУ БиоАПИ.

Handle (входной) — дескриптор извлекаемой ЗБИ.

BIR (выходной) — указатель на извлеченную ЗБИ.

8.2.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.2.2.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE

BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.2.3 Операция BioAPI_GetHeaderFromHandle

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

8.2.3.1 Описание

Данная функция извлекает заголовок ЗБИ, связанной с данным дескриптором. ПБУ не освобождает дескриптор ЗБИ.

8.2.3.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

Handle (входной) — дескриптор ЗБИ, заголовок которой должен быть извлечен.

Header (выходной) — заголовок указанной ЗБИ.

8.2.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.2.3.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE

BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3 Обратные вызовы и работа с событиями

8.3.1 Функция BioAPI_EnableEvents

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

8.3.1.1 Описание

Данная функция разрешает события, указанные в маске событий, исходящие от всех модулей БиоАПИ, выбранных в прикрепленной сессии ПБУ, идентифицированной дескриптором ПБУ, и отключает все другие события от этих модулей БиоАПИ. События от других модулей БиоАПИ, непосредственно или косвенно управляемых тем же ПБУ (которые могут быть выбраны в другой прикрепленной сессии и не выбраны в данной), не обрабатываются.

В некоторых случаях событие «вставка» не может быть заблокировано.

8.3.1.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

Events (входной) — маска, обозначающая разрешенные события.

8.3.1.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.1.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.2 Функция BioAPI_SetGUICallbacks

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

8.3.2.1 Описание

Данная функция позволяет приложению устанавливать обратные вызовы так, что приложение может управлять внешним видом и поведением биометрического интерфейса пользователя, получая от ПБУ последовательные битовые изображения, называемые потоковыми данными, для их отображения биометрическим приложением, а также информацию о режиме.

Примечание — Не все ПБУ поддерживают функцию предоставления потоковых данных.

8.3.2.2 Параметры

BSPHandle (входной) — дескриптор присоединенного поставщика биометрической услуги .

GuiStreamingCallback (входной) — указатель на обратный вызов приложения для взаимодействия с представлением потоковых биометрических данных.

GuiStreamingCallbackCtx (входной) — указатель на контекстную информацию, предоставленную приложением, которая будет представлена в обратном вызове.

GuiStateCallback (входной) — указатель на обратный вызов приложения для взаимодействия с изменениями состояния ГИП.

GuiStateCallbackCtx (входной) — указатель на контекстную информацию, предоставленную приложением, которая будет представлена в обратном вызове.

Примечание 1 — Определения функциональных подтипов для BioAPI_GUI_STATE_CALLBACK и BioAPI_GUI_STREAMING_CALLBACK приведены в 7.36 и 7.37 соответственно.

Примечание 2 — Определение интерфейса пользователя приведено в приложении С, раздел С.7.

8.3.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают условия ошибки.

8.3.2.4 Ошибки

BioAPIERR_INVALID_POINTER
BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4 Биометрические функции

8.4.1 Функция BioAPI_Capture

```
BioAPI_RETURN BioAPI BioAPI_Capture
  (BioAPI_HANDLE BSPHandle,
  BioAPI_BIR_PURPOSE Purpose,
  BioAPI_BIR_SUBTYPE Subtype,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
  BioAPI_BIR_HANDLE *CapturedBIR,
  int32_t Timeout,
  BioAPI_BIR_HANDLE *AuditData);
```

8.4.1.1 Описание

Данная функция получает образцы для указанного назначения, ПБУ возвращает ЗБИ «промежуточного» типа в случае, если после этого необходимо вызывать функцию ***BioAPI_Process,*** или «обработанную» ЗБИ в противном случае. Назначение записано в заголовке CapturedBIR. Если AuditData является ненулевым указателем, может быть возвращен ЗБИ «исходного» типа. Функция возвращает дескрипторы

к любым собранным данным, и все локальные операции могут быть выполнены с помощью дескрипторов. Если приложению необходимо получить данные для их сохранения в базе данных ЗБИ или пересылки на сервер, оно может извлечь их путем вызова функции ***BioAPI_GetBIRFromHandle.***

По умолчанию за предоставление связанного с получением данных интерфейса пользователя отвечает ПБУ. Приложение может запросить управление видом и поведением ГИП путем предоставления указателя обратного вызова ГИП в BioAPI_SetGUICallbacks. Дополнительное описание особенностей интерфейса пользователя приведено в приложении С, раздел С.7.

Использование устройств получения биометрических данных происходит последовательно. Если два или более биометрических приложения одновременно запрашивают датчик, «опоздавшие» будут ждать завершения операции или истечения времени ожидания. Этот порядок установлен во всех функциях получения данных. За сериализацию отвечает ПБУ. В свою очередь ПБУ реализует сериализацию, используя признак «занято» (BioAPI_UNIT_IN_USE) или устанавливая очередность.

Дескриптор ЗБИ, возвращаемый функцией, должен быть освобожден приложением путем вызова функции ***BioAPI_FreeBIRHandle,*** если приложение больше не использует его.

8.4.1.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

Purpose (входной) — значение, указывающее назначение получаемых биометрических данных.

Subtype (входной/необязательный) — определяет тип полученных данных (например, левый или правый глаз). Значение BioAPI_NO_SUBTYPE_AVAILABLE (0x00) показывает, что ПБУ должен выбрать подтип данных.

Примечание — Не все ПБУ поддерживают получение специфических данных. Действительный подтип должен быть указан в заголовке возвращаемого *CapturedBIR*.

OutputFormat (входной/необязательный) — определяет формат ББД, который будет использоваться при возвращении *CapturedBIR* в том случае, если ПБУ поддерживает более одного формата. Пустой указатель означает, что ПБУ должен выбрать формат.

CapturedBIR (выходной) — дескриптор ЗБИ, содержащей полученные данные. Эти данные являются либо ЗБИ «промежуточного» типа (которая может использоваться исключительно функциями ***BioAPI_Process, BioAPI_CreateTemplate*** или ***BioAPI_ProcessWithAuxData*** в зависимости от назначения), либо «обработанной» ЗБИ (которая может непосредственно использоваться функциями ***BioAPI_VerifyMatch*** или ***BioAPI_IdentifyMatch*** в зависимости от назначения).

Timeout (входной) — целое число, определяющее значение времени ожидания (в миллисекундах) для операции. Если время ожидания истекло, функция возвращает ошибку без предоставления результатов операции. Данное значение может быть любым положительным числом. Значение минус 1 означает, что будет использоваться значение времени ожидания, заданное по умолчанию ПБУ.

AuditData (выходной / необязательный) — дескриптор ЗБИ, содержащей исходные биометрические данные. Эти данные могут использоваться для предоставления биометрических данных личности. Нулевой указатель на входе указывает, что контрольные данные не будут собраны. Не все ПБУ поддерживают сбор контрольных данных. ПБУ может возвратить значение дескриптора BioAPI_UNSUPPORTED_BIR_HANDLE, чтобы указать, что *AuditData* не поддерживается, или значение BioAPI_INVALID_BIR_HANDLE, чтобы указать, что контрольные данные недоступны.

8.4.1.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.4.1.4 Ошибки

BioAPIERR_USER_CANCELLED
BioAPIERR_UNABLE_TO_CAPTURE
BioAPIERR_TOO_MANY_HANDLES
BioAPIERR_TIMEOUT_EXPIRED
BioAPIERR_PURPOSE_NOT_SUPPORTED
BioAPIERR_UNSUPPORTED_FORMAT
BioAPIERR_UNIT_IN_USE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4.2 Функция BioAPI_CreateTemplate

```
BioAPI_RETURN BioAPI BioAPI_CreateTemplate
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_INPUT_BIR *CapturedBIR,
```

const BioAPI_INPUT_BIR *ReferenceTemplate,
const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
BioAPI_BIR_HANDLE *NewTemplate,
const BioAPI_DATA *Payload,
BioAPI_UUID *TemplateUUID);

8.4.2.1 Описание

Данная функция принимает ЗБИ, содержащую биометрические данные в промежуточной форме, для создания нового шаблона регистрации. Новая ЗБИ создается с помощью CapturedBIR. Также функция может дополнительно выполнить адаптацию на основе существующего ReferenceTemplate (контрольного шаблона). Старый ReferenceTemplate остается неизменным.

Дополнительный входной ReferenceTemplate предоставляется для использования при создании NewTemplate (нового шаблона), если ПБУ поддерживает эту возможность. Использование ПБУ входного ReferenceTemplate для создания выходного NewTemplate необязательно.

Если ПБУ поддерживает внутреннюю или управляемую ПБУ базу данных ЗБИ (например, смарт-карту или механизм идентификации), то он может дополнительно возвратить УУИД, присвоенный вновь созданному ReferenceTemplate, сохраненному в управляемой ПБУ базе данных ЗБИ. Значение УУИД должно быть таким же, как и значение, включенное в заголовок ЗБИ, если оно там присутствует.

Дескриптор ЗБИ, возвращаемый приложением, должен быть освобожден приложением BioAPI_GetBIRFromHandle, если он больше не использует его. ЗБИ может быть восстановлена путем вызова функции ***BioAPI_GetBIRFromHandle,*** которая освобождает дескриптор.

8.4.2.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

CapturedBIR (входной) — полученная ЗБИ или ее дескриптор.

ReferenceTemplate (входной/необязательный) — необязательно существующий шаблон, который должен быть адаптирован, или его ключ в базе данных ЗБИ, или его дескриптор.

OutputFormat (входной/необязательный) — определяет, какой формат ББД будет использоваться при возвращении NewTemplate, если ПБУ поддерживает более одного формата. Пустой указатель означает, что формат должен быть выбран ПБУ.

NewTemplate (выходной) — дескриптор вновь созданного шаблона, который получен из параметра *CapturedBIR* с использованием (необязательно) *ReferenceTemplate*.

Payload (входной/необязательный) — указатель на данные, которые будут сохранены ПБУ. Данный параметр игнорируется, если указатель пустой.

Примечание 1 — Не все ПБУ поддерживают сохранение полезной информации.

Примечание 2 — Дополнительно о полезной информации указано в приложении А, подпункт А.4.6.2.6, и приложении С, раздел С.5.

TemplateUUID (выходной/необязательный) — указатель на 16-байтовую область памяти, в которую дополнительно может быть возвращен присвоенный ПБУ УУИД, связанный с *ReferenceTemplate* (сохраненным в управляемой ПБУ базе данных ЗБИ). Если не требуется возвращения УУИД, указатель должен быть пустым.

8.4.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.4.2.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_INVALID_BIR
BioAPIERR_BIR_SIGNATURE_FAILURE
BioAPIERR_TOO_MANY_HANDLES
BioAPIERR_UNABLE_TO_STORE_PAYLOAD
BioAPIERR_INCONSISTENT_PURPOSE
BioAPIERR_PURPOSE_NOT_SUPPORTED
BioAPIERR_UNSUPPORTED_FORMAT
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_QUALITY_ERROR

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4.3 Функция BioAPI_Process

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

8.4.3.1 Описание

Данная функция обрабатывает промежуточные данные, полученные путем вызова функции ***BioAPI_Capture,*** для последующей верификации или идентификации. Если присоединенный вызов ПБУ позволяет проводить обработку, то ПБУ создает обработанный биометрический образец ЗБИ. В противном случае ProcessedBIR приравнивается к пустому указателю и функция возвращает BioAPIERR_BSP_FUNCTION_NOT_SUPPORTED.

Данная функция приводит к созданию ЗБИ ПБУ. Приложение может восстановить ЗБИ с помощью дескриптора путем вызова функции ***BioAPI_GetBIRFromHandle,*** которая также освобождает дескриптор или может освободить память, связанную с дескриптором ЗБИ путем вызова функции ***BioAPI_FreeBIRHandle.***

8.4.3.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

CapturedBIR (входной) — полученная ЗБИ или ее дескриптор.

OutputFormat (входной/необязательный) — определяет, какой формат ББД будет использоваться при возвращении ProcessedBIR, если ПБУ поддерживает более одного формата. Пустой указатель означает, что формат должен быть выбран ПБУ.

ProcessedBIR (выходной) — дескриптор вновь созданной «обработанной» ЗБИ должен быт равен пустому указателю.

8.4.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнении функции. Все остальные значения описывают тип ошибки.

8.4.3.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_INVALID_BIR
BioAPIERR_BIR_SIGNATURE_FAILURE
BioAPIERR_TOO_MANY_HANDLES
BioAPIERR_INCONSISTENT_PURPOSE
BioAPIERR_PURPOSE_NOT_SUPPORTED
BioAPIERR_UNSUPPORTED_FORMAT
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_QUALITY_ERROR

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4.4 Функция BioAPI_ProcessWithAuxBIR

```
BioAPI_Return BioAPI BioAPI_ProcessPrematchData
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_INPUT_BIR *CapturedBIR,
  const BioAPI_INPUT_BIR *PrematchData,
  BioAPI_HANDLE_PTR ProcessedBIR);
```

8.4.4.1 Описание

Данная функция обрабатывает промежуточные данные, полученные при вызове функции ***BioAPI_Capture***, в соответствии со вспомогательными данными, создавая обработанные биометрические образцы с целью их последующей верификации или идентификации. Это позволяет разрабатывать приложения, требующие в виде входного параметра вспомогательные данные обработки.

Примечание — Данная функция может использоваться для поддержки биометрических сопоставлений-на-карте (СНК). Описание использования БиоАПИ для полной обработки СНК, приведено в приложении С, раздел С.8.

Если возможность работы со вспомогательными данными поддерживается прикрепленным обращением ПБУ, то ПБУ создает обработанный биометрический образец ЗБИ, иначе, параметр ProcessedBIR установлен в ноль, и функция возвращает BioAPIERR_BSP_FUNCTION_NOT_SUPPORTED.

Использование данной функции приводит к созданию ЗБИ ПБУ. Приложение может восстановить ЗБИ с помощью дескриптора путем вызова функции ***BioAPI_GetBIRFromHandle,*** которая также освобождает дескриптор, или может освободить память, связанную с дескриптором ЗБИ путем вызова функции ***BioAPI_FreeBIRHandle.***

8.4.4.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

CapturedBIR (входной) — ЗБИ, полученная в результате предварительно вызванной функции ***BioAPI_Capture.***

AuxiliaryData (входной) — ЗБИ, содержащая вспомогательные данные, используемые для операций обработки.

Примечание 1 — Примером вспомогательных данных может служить информация, связанная с регистрацией шаблона, которая дает возможность обработчику должным образом выполнить сжатие входного изображения, чтобы максимизировать возможность последующего сопоставления (например, для гарантии того, что обработанная ЗБИ для верификации и зарегистрированный шаблон выделены в одной и той же части пальца).

Примечание 2 — Содержание и формат вспомогательных данных определены полем формата биометрических данных ЗБИ в заголовке вспомогательной ЗБИ и могут быть специфичными для каждого ПБУ.

OutputFormat (входной/необязательный) — определяет, какой формат ББД будет использоваться при возвращении ProcessedBIR, если ПБУ поддерживает более одного формата. Пустой указатель означает, что формат должен выбрать ПБУ.

ProcessedBIR (выходной) — дескриптор для вновь созданной «обработанной» ЗБИ.

8.4.4.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.4.4.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_INVALID_BIR
BioAPIERR_BIR_SIGNATURE_FAILURE
BioAPIERR_TOO_MANY_HANDLES
BioAPIERR_INCONSISTENT_PURPOSE
BioAPIERR_PURPOSE_NOT_SUPPORTED
BioAPIERR_UNSUPPORTED_FORMAT
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_FUNCTION_NOT_SUPPORTED

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4.5 Функция BioAPI_VerifyMatch

```
BioAPI_RETURN BioAPI BioAPI_VerifyMatch
  (BioAPI_HANDLE BSPHandle,
  BioAPI_FMR MaxFMRRequested,
  const BioAPI_INPUT_BIR *ProcessedBIR,
  const BioAPI_INPUT_BIR *ReferenceTemplate,
  BioAPI_BIR_HANDLE *AdaptedBIR,
  BioAPI_BOOL *Result,
  BioAPI_FMR *FMRAchieved,
  BioAPI_DATA *Payload);
```

8.4.5.1 Описание

Данная функция выполняет верификацию, т. е. сопоставление «один к одному» двух ЗБИ: ProcessedBIR и ReferenceTemplate. ProcessedBIR представляет собой «обработанную» ЗБИ, специально созданную для проведения верификации. ReferenceTemplate (контрольный шаблон) создается при регистрации.

Приложение должно запросить максимальное значение критерия (порога) ОЛС для успешного выполнения верификации. Булево значение *Result* указывает, была ли верификация проведена успешно или нет, а FMRAchieved представляет собой значение ОЛС, указывающее степень близости сопоставляемых ЗБИ.

Примечание —Дополнительная информация о понятии ОЛС для нормализованного оценивания схожести и выбора порога приведена в приложении С, раздел С.4.

Установкой указателя AdaptedBIR на непустой указатель приложение может потребовать, чтобы ЗБИ была создана адаптацией ReferenceTemplate с использованием ProcessedBIR. Новый дескриптор возвращается указателем AdaptedBIR. Если сопоставление проведено успешно, то может быть сделана попытка адаптации ReferenceTemplate с использованием информации, взятой из ProcessedBIR. Не все ПБУ поддерживают адаптацию. Полученный AdaptedBIR должен считаться оптимальным шаблоном регистрации и должен быть сохранен в базе данных регистрации. Решение об использовании или отказе от использования этих данных принимает приложение. Важно отметить, что адаптация может быть проведена не во всех случаях. При выполнении адаптации данная функция сохраняет дескриптор новой ЗБИ в памяти, на которую указывает параметр AdaptedBIR.

Если ReferenceTemplate связана с Payload (полезная информация), то она может быть возвращена при условии успешной верификации, если FMRAchieved имеет допустимое значение; данный процесс управляется политикой ПБУ и должен быть указан в его схеме.

Примечание 1 — Не все ПБУ поддерживают возвращение полезной информации.

Примечание 2 — Дополнительная информация об использовании Payload приведена в приложении А, подпункт А.4.6.2.6, и приложении С, раздел С.5.

Область памяти, возвращаемая при вызове функции БиоАПИ, должна быть освобождена приложением сразу же после прекращения его использования функцией ***BioAPI_Free*** (см. 8.7.2). Если адаптированная ЗБИ возвращается, ее дескриптор может быть освобожден путем вызова функции ***BioAPI_FreeBIRHandle.***

8.4.5.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

MaxFMRRequested (входной) — значение ОЛС, являющееся критерием успешной верификации (то есть порогом сопоставления).

ProcessedBIR (входной) — верифицируемая ЗБИ или ее дескриптор.

ReferenceTemplate (входной) — верифицирующая (контрольная) ЗБИ, ее ключ в базе данных ЗБИ или ее дескриптор.

AdaptedBIR (выходной/необязательный) — указатель на дескриптор адаптированной ЗБИ. Данный указатель может быть пустым, если не требуется создание адаптированной ЗБИ. Не все ПБУ поддерживают адаптацию ЗБИ. Функция может возвратить значение дескриптора BioAPI_UNSUPPORTED_BIR_HANDLE, чтобы указать, что адаптация не поддерживается, или значение BioAPI_INVALID_BIR_HANDLE, чтобы указать, что адаптация невозможна.

Result (выходной) — указатель на Булево значение (BioAPI_TRUE/BioAPI_FALSE), показывающее, является ли результат сопоставления ЗБИ положительным или нет в соответствии с установленными критериями.

FMRAchieved (выходной) — указатель на значение ОЛС, означающее близость соответствия.

Payload (выходной/необязательный) — если с *ReferenceTemplate* связана полезная информация, то она возвращается в выделенную в памяти структуру BioAPI_DATA в том случае, если значение *FMRAchieved* соответствует политике ПБУ.

8.4.5.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.4.5.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_INVALID_BIR
BioAPIERR_BIR_SIGNATURE_FAILURE
BioAPIERR_INCONSISTENT_PURPOSE
BioAPIERR_BIR_NOT_FULLY_PROCESSED
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_QUALITY_ERROR
Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4.6 Функция BioAPI_IdentifyMatch

```
BioAPI_RETURN BioAPI BioAPI_IdentifyMatch
  (BioAPI_HANDLE BSPHandle,
  BioAPI_FMR MaxFMRRequested,
  const BioAPI_INPUT_BIR *ProcessedBIR,
```

```
const BioAPI_IDENTIFY_POPULATION *Population,
uint32_t TotalNumberOfTemplates,
BioAPI_BOOL Binning,
uint32_t MaxNumberOfResults,
uint32_t *NumberOfResults,
BioAPI_CANDIDATE **Candidates,
int32_t Timeout);
```

8.4.6.1 Описание

Данная функция выполняет идентификацию, т. е. сопоставление «один ко многим» ProcessedBIR и набора контрольных ЗБИ. ProcessedBIR представляет собой «обработанную» ЗБИ, полученную специально для проведения идентификации. Совокупность, с которой проводится сопоставление, может быть предоставлена одним из двух способов:

a) в базе данных ЗБИ, обозначаемой открытым дескриптором базы данных;

b) во входном параметре в виде массива ЗБИ.

Примечание — При использовании управляемой БПУ базы данных ЗБИ, она должна быть сначала открыта путем вызова функции ***BioAPI_DbOpen.***

Существует возможность использования массива ЗБИ, который может быть определен в BioAPI_IDENTIFY_POPULATION_TYPE в структуре BioAPI_IDENTIFY_POPULATION. Если он определен как BioAPI_PRESET_ARRAY_TYPE (3), то будет использоваться массив ЗБИ, который был предварительно установлен при вызове функции ***BioAPI_PresetIdentifyPopulation.*** Предварительно установленный массив ЗБИ будет освобожден самим ПБУ при вызове функции ***BioAPI_BSPDetach.***

Функция выполняется в следующем порядке.

a) определяет число кандидатов из популяции, чьи сопоставления соответствуют определенному критерию;

b) выделяет область памяти, достаточную для размещения массива элементов типа BioAPI_CANDIDATE, с числом элементов, равным числу кандидатов, определенных в перечислении a);

c) заполняет массив информацией о кандидатах, определенных в перечислении a), включая FMRAchieved каждого кандидата;

d) возвращает адрес массива в параметре *Candidates* и размер массива в параметре *NumberOfResults*.

Примечание — Дополнительная информация об использовании понятия ОЛС для нормализованного оценивания схожести и выбора порога приведена в приложении С, раздел С.4.

Блок памяти, возвращаемый путем вызова функции БиоАПИ, должен быть освобожден приложением путем вызова функции ***BioAPI_Free*** (см. 8.7.2).

8.4.6.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

MaxFMRRequested (входной) — значение, являющееся критерием успешной идентификации (то есть порогом сопоставления).

ProcessedBIR (входной) — идентифицируемая ЗБИ.

Population (входной) — совокупность контрольных шаблонов ЗБИ, по которым проводится идентификация (с помощью ПБУ).

TotalNumberOfTemplates (входной) — определяет общее число шаблонов сохраненной приложением совокупности. Нулевое значение указывает, что приложение не предоставляет число шаблонов.

Примечание — Если совокупность распределена по нескольким базам данным/частям, то общий размер совокупности будет больше размера той совокупности, которую видит ПБУ. ПБУ может отображать *FARRequested* на свой внутренний порог сопоставления, основываясь на этом общем размере совокупности.

Binning (входной) — Булево значение, указывающее, включен или нет режим категоризации.

Примечание 1 — Категоризация — методика оптимизации поиска, которую может использовать ПБУ. Данная методика основана на поиске подмножества совокупности по внутренним характеристикам биометрических данных. Наряду с увеличением скорости операции сопоставления может также увеличиться вероятность пропуска кандидата из-за возможной ошибочной буферизации и как результат обнаружения буфера, который должен содержать, но не содержит сопоставляемую ЗБИ.

Примечание 2 — Дополнительная информация о категоризации приведена в приложении А, подпункт А.4.6.2.10.

MaxNumberOfResults (входной) — определяет максимальное число кандидатов, возвращаемых в результате сопоставления 1:N. Нулевое значение указывает на необходимость возврата всех кандидатов.

NumberOfResults (выходной) — указатель на число кандидатов, возвращенных в параметре *Candidates* в качестве результата сопоставления 1:N.

Candidates (выходной) — указатель на адрес массива элементов типа BioAPI_CANDIDATE, содержащий информацию о ЗБИ, идентифицированных как результат процесса сопоставления (то есть индексы, связанные с ЗБИ, для которых был превышен порог сопоставления). Данный массив является ранжированным, причем запись с лучшей оценкой схожести (наиболее соответствующая) находится на первом месте. Если ни одного кандидата не найдено, память под массив не выделяется и возвращается пустой указатель. Если параметр Population был представлен в базе данных ЗБИ, т. е. значение BioAPI_IDENTIFY_POPULATION_TYPE равно BioAPI_DB_TYPE, то массив содержит указатели на УУИД, связанные с ЗБИ, хранящимися во внутренней базе данных регистрации ПБУ. Если параметр *Population* был представлен как массив ЗБИ, то BioAPI_IDENTIFY_POPULATION_TYPE имеет значение BioAPI_ARRAY_TYPE и массив *Candidates* содержит указатели на относительные индексы в передаваемом массиве.

TimeOut (входной) — целое число, определяющее значение времени ожидания (в миллисекундах) для выполнения операции. Если время ожидания истекло, функция возвращает ошибку, память для массива не выделяется и возвращается пустой указатель. Данное значение может быть любым положительным числом. Значение минус 1 означает, что будет использоваться значение времени ожидания, заданное ПБУ по умолчанию.

8.4.6.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.4.6.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_BIR_SIGNATURE_FAILURE
BioAPIERR_TIMEOUT_EXPIRED
BioAPIERR_NO_INPUT_BIRS
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_INCONSISTENT_PURPOSE
BioAPIERR_BIR_NOT_FULLY_PROCESSED
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_QUALITY_ERROR
BioAPIERR_FUNCTION_FAILED
BioAPIERR_PRESET_BIR_DOES_NOT_EXIST
BioAPIERR_INVALID_DB_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

Примечание 1 — Не все ПБУ поддерживают идентификацию 1:N. Данное указание должно быть приведено в руководстве по программированию конкретного ПБУ.

Примечание 2 — В зависимости от ПБУ, местоположения и размера базы данных, по которой производится поиск, выполнение данной операции может занять значительное время. Рекомендуемое значение параметра *Timeout* должно быть указано в руководстве по программированию конкретного ПБУ.

Примечание 3 — Число кандидатов сопоставления, найденных ПБУ, зависит от текущего ОЛС алгоритма сопоставления, установленного в параметре MaxFMRRequested.

8.4.7 Функция BioAPI_Enroll

```
BioAPI_RETURN BioAPI BioAPI_Enroll
  (BioAPI_HANDLE BSPHandle,
  BioAPI_BIR_PURPOSE Purpose,
  BioAPI_BIR_SUBTYPE Subtype,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
  const BioAPI_INPUT_BIR *ReferenceTemplate,
  BioAPI_BIR_HANDLE *NewTemplate,
  const BioAPI_DATA *Payload,
  int32_t Timeout,
  BioAPI_BIR_HANDLE *AuditData,
  BioAPI_UUID *TemplateUUID);
```

8.4.7.1 Описание

Данная функция получает биометрические данные от присоединенного устройства (модуля датчика) для создания *ProcessedBIR* с целью создания BioAPI_PURPOSE_ENROLL, BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY или BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION ONLY (то есть для создания контрольного шаблона).

Необязательный входной параметр *ReferenceTemplate* используется при создании нового шаблона *NewTemplate*, если ПБУ поддерживает возможность обновления шаблона. При этом использование входного шаблона *ReferenceTemplate* ПБУ для создания выходного шаблона *NewTemplate* необязательно.

Если ПБУ поддерживает внутреннюю (или управляемую ПБУ) базу данных ЗБИ (например, смарт-карту или механизм идентификации), то он может дополнительно возвратить УУИД, присвоенный вновь созданному *ReferenceTemplate* (контрольному шаблону), сохраненному в базе данных ЗБИ, управляемой ПБУ. Значение УУИД должно быть таким же, как и значение, включенное в заголовок ЗБИ, если оно там присутствует.

По умолчанию за предоставление связанного с операцией регистрации интерфейса пользователя отвечает ПБУ. Приложение может запросить управление видом и поведением ГИП в виде предоставления указателя обратного вызова ГИП в BioAPI_SetGUICallbacks. Дополнительное описание особенностей интерфейса пользователя приведено в приложении С, раздел С.7.

Так как функция ***BioAPI_Enroll*** включает в себя функцию получения данных, она упорядочивает использование устройства получения биометрических данных. Если два или более биометрических приложения одновременно запрашивают датчик, «опоздавшие» должны будут ждать завершения операции или истечения времени ожидания. Этот порядок установлен во всех функциях получения данных. Сериализацию осуществляет ПБУ. Это может быть реализовано либо возвращением ошибки «занято» (BioAPI_UNIT_IN_USE), либо организацией очередности.

Область памяти, возвращаемая вызовом функции БиоАПИ, должна быть освобождена приложением путем вызова функции ***BioAPI_Free,*** если приложение больше не будет ее использовать (см. 8.7.2). Выходные ЗБИ могут быть восстановлены путем вызова функции ***BioAPI_GetBIRFromHandle,*** которая освобождает дескриптор, или дескриптор может быть освобожден без восстановления ЗБИ путем вызова функции ***BioAPI_FreeBIRHandle.***

8.4.7.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

Purpose (входной) — значение, указывающее желаемый подтип получаемых данных (BioAPI_PURPOSE_ENROLL, BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY или BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION ONLY).

Subtype (входной/необязательный) — определяет подтип регистрации (например, левый или правый глаз). Значение BioAPI_NO_SUBTYPE_AVAILABLE (0x00) указывает, что подтип должен выбрать ПБУ.

Примечание — Не все ПБУ поддерживают получение данных определенного подтипа. Используемый подтип получения данных будет отображен в заголовке возвращаемого параметра *NewTemplate*.

OutputFormat (входной/необязательный) — определяет формат ББД возвращаемого *NewTemplate*, если ПБУ поддерживает более одного формата. Пустой указатель означает, что ПБУ должен выбрать формат.

ReferenceTemplate (входной/необязательный) — адаптируемая (обновляемая) ЗБИ, ее ключ в базе данных или ее дескриптор.

NewTemplate (выходной) — дескриптор вновь созданного шаблона, который получен из новых исходных образцов и (необязательно) из *ReferenceTemplate*.

Payload (входной/необязательный) — указатель на данные, которые будут сохранены ПБУ. Данный параметр игнорируется, если указатель пустой.

Примечание 1 — Не все ПБУ поддерживают хранение полезной информации.

Примечание 2 — Дополнительная информация о полезной информации, приведена в приложении А, подпункт А.4.6.2.6, и приложении С, раздел С.5.

Timeout (входной) — целое число, определяющее значение времени ожидания (в миллисекундах) для операции. Если время ожидания истекло, функция возвращает ошибку без результатов операции. Данное значение может быть любым положительным числом. Значение минус 1 означает, что будет использоваться значение времени ожидания, заданное ПБУ по умолчанию.

AuditData (выходной/необязательный) — дескриптор ЗБИ, содержащей контрольные биометрические данные, которые могут использоваться для предоставления биометрических данных личности. Нуле-

вой указатель на входе указывает, что контрольные данные не будут собраны. Не все ПБУ поддерживают сбор контрольных данных. ПБУ может возвратить значение дескриптора BioAPI_UNSUPPORTED_BIR_HANDLE для того, чтобы указать, что *AuditData* не поддерживается, или значение BioAPI_INVALID_BIR_HANDLE, чтобы указать, что контрольные данные недоступны.

TemplateUUID (выходной/необязательный) — указатель на 16-байтовый блок памяти, в который будет возвращен (необязательно) назначенный ПБУ УУИД, связанный с *ReferenceTemplate* (сохраненным в управляемой ПБУ базе данных ЗБИ). Указатель должен быть нулевым, если УУИД не возвращается.

8.4.7.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.4.7.4 Ошибки

BioAPIERR_USER_CANCELLED
BioAPIERR_UNABLE_TO_CAPTURE
BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_TOO_MANY_HANDLES
BioAPIERR_UNABLE_TO_STORE_PAYLOAD
BioAPIERR_TIMEOUT_EXPIRED
BioAPIERR_PURPOSE_NOT_SUPPORTED
BioAPIERR_UNSUPPORTED_FORMAT
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_QUALITY_ERROR
BioAPIERR_UNIT_IN_USE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4.8 Функция BioAPI_Verify

```
BioAPI_RETURN BioAPI BioAPI_Verify
  (BioAPI_HANDLE BSPHandle,
  BioAPI_FMR MaxFMRRequested,
  const BioAPI_INPUT_BIR *ReferenceTemplate,
  BioAPI_BIR_SUBTYPE Subtype,
  BioAPI_BIR_HANDLE *AdaptedBIR,
  BioAPI_BOOL *Result,
  BioAPI_FMR *FMRAchieved,
  BioAPI_DATA *Payload,
  int32_t Timeout,
  BioAPI_BIR_HANDLE *AuditData);
```

8.4.8.1 Описание

Данная функция получает биометрические данные от присоединенного устройства (модуля датчика) и сравнивает их с контрольным шаблоном *ReferenceTemplate*.

Приложение должно запросить максимальное значение критерия ОЛС (пороговое значение) для успешного сопоставления. Булево значение *Result* указывает, проведена верификация успешно или нет, а параметр *FMRAchieved* представляет собой значение ОЛС, указывающее степень близости сопоставляемых ЗБИ.

Примечание — Дополнительная информация об использовании понятия ОЛС для нормализованного оценивания схожести и выбора порога приведена в приложении С, раздел С.4.

Устанавливая указатель *AdaptedBIR*, приложение может потребовать, чтобы ЗБИ была сформирована адаптацией *ReferenceTemplate* с использованием ProcessedBIR. Новый дескриптор должен быть возвращен в *AdaptedBIR*. Если сопоставление проведено успешно, может быть сделана попытка адаптации *ReferenceTemplate* с использованием информации, взятой из *ProcessedBIR*. (Не все ПБУ поддерживают адаптацию). Полученный *AdaptedBIR* считают оптимальным шаблоном регистрации, и он должен быть сохранен в базе данных регистрации (решение об использовании или отказе от использования этих данных принимает приложение). Важно отметить, что адаптация может быть проведена не во всех случаях. При выполнении адаптации данная функция хранит дескриптор новой ЗБИ в памяти, на которую указывает параметр *AdaptedBIR*.

Если с *ReferenceTemplate* связана *Payload* (полезная информация), то она может быть возвращена при условии успешной верификации, если *FMRAchieved* достаточно строгий, что контролируется политикой ПБУ и определено в его схеме.

Примечание 1 — Не все ПБУ возвращают полезную информацию.

Примечание 2 — Дополнительная информация относительно использования параметра Payload приведена в приложении А, подпункт А.4.6.2.6, и приложении С, раздел С.5.

По умолчанию за предоставление связанного с операцией верификации интерфейса пользователя отвечает ПБУ. Приложение может запросить управление видом и поведением ГИП в виде предоставления указателя обратного вызова ГИП в функции ***BioAPI_SetGUICallbacks.*** Дополнительное описание особенностей интерфейса пользователя приведено в приложении С, раздел С.7.

Так как функция ***BioAPI_Verify*** включает в себя операцию получения данных, она упорядочивает использование устройств получения биометрических данных. Если два или более биометрических приложения одновременно запрашивают датчик, «опоздавшие» приложения должны будут ждать завершения операции или истечения времени ожидания. Этот порядок установлен во всех функциях получения данных. Сериализацию осуществляет ПБУ. Это может быть реализовано либо возвращением ошибки «занято» (BioAPI_UNIT_IN_USE), либо организацией очередности.

Область памяти, возвращаемая вызовом функции БиоАПИ, должна быть освобождена приложением путем вызова функции ***BioAPI_Free,*** если приложение больше не будет ее использовать (см. 8.7.2). Выходные ЗБИ могут быть восстановлены путем вызова функции ***BioAPI_GetBIRFromHandle,*** которая освобождает дескриптор, или дескриптор может быть освобожден без восстановления ЗБИ путем вызова функции ***BioAPI_FreeBIRHandle.***

8.4.8.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

MaxFMRRequested (входной) — значение ОЛС, являющееся критерием успешной верификации (порогом сопоставления).

ReferenceTemplate (входной) — верифицирующая (контрольная) ЗБИ, ее ключ в базе данных ЗБИ или ее дескриптор.

Subtype (входной / необязательный) — определяет подтип регистрации (например, левый или правый глаз). Значение BioAPI_NO_SUBTYPE_AVAILABLE (0x00) указывает, что подтип должен выбрать ПБУ.

Примечание — Не все ПБУ поддерживают получение данных определенного подтипа.

AdaptedBIR (выходной/необязательный) — указатель на дескриптор адаптированной ЗБИ. Данный указатель может быть пустым, если не требуется создание адаптированной ЗБИ. Не все ПБУ поддерживают адаптацию ЗБИ. Функция может возвратить значение дескриптора BioAPI_UNSUPPORTED_BIR_HANDLE, чтобы указать, что адаптация не поддерживается, или значение BioAPI_INVALID_BIR_HANDLE, чтобы указать, что адаптация невозможна.

Result (выходной) — указатель на Булево значение (BioAPI_TRUE / BioAPI_FALSE), показывающее, является ли результат сопоставления ЗБИ положительным или нет в соответствии с установленными критериями.

FMRAchieved (выходной) — указатель на значение ОЛС, означающее близость соответствия (оценка схожести).

Payload (выходной/необязательный) — если с *ReferenceTemplate* связана полезная информация, то она возвращается в выделенную в памяти структуру BioAPI_DATA в том случае, если значение *FARAchieved* соответствует политике ПБУ.

Timeout (входной) — целое число, определяющее значение времени ожидания (в миллисекундах) для операции. Если время ожидания истекло, функция возвращает ошибку без предоставления результатов операции. Данное значение может быть любым положительным числом. Значение минус 1 означает, что будет использоваться значение времени ожидания, заданное по умолчанию ПБУ.

AuditData (выходной / необязательный) — дескриптор ЗБИ, содержащей исходные биометрические данные. Эти данные могут использоваться для предоставления биометрических данных личности. Нулевой указатель на входе указывает, что контрольные данные не должны быть собраны. Не все ПБУ поддерживают сбор контрольных данных. ПБУ может возвратить значение дескриптора BioAPI_UNSUPPORTED_BIR_HANDLE, чтобы указать, что *AuditData* не поддерживается, или значение BioAPI_INVALID_BIR_HANDLE, чтобы указать, что контрольные данные недоступны.

8.4.8.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.4.8.4 Ошибки

BioAPIERR_USER_CANCELLED
BioAPIERR_UNABLE_TO_CAPTURE
BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_BIR_SIGNATURE_FAILURE
BioAPIERR_TOO_MANY_HANDLES
BioAPIERR_TIMEOUT_EXPIRED
BioAPIERR_INCONSISTENT_PURPOSE
BioAPIERR_UNSUPPORTED_FORMAT
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_QUALITY_ERROR
BioAPIERR_UNIT_IN_USE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4.9 Функция BioAPI_Identify

```
BioAPI_RETURN BioAPI BioAPI_Identify
  (BioAPI_HANDLE BSPHandle,
  BioAPI_FMR MaxFMRRequested,
  BioAPI_BIR_SUBTYPE Subtype,
  const BioAPI_IDENTIFY_POPULATION *Population,
  uint32_t TotalNumberOfTemplates,
  BioAPI_BOOL Binning,
  uint32_t MaxNumberOfResults,
  uint32_t *NumberOfResults,
  BioAPI_CANDIDATE **Candidates,
  int32_t Timeout,
  BioAPI_BIR_HANDLE *AuditData);
```

8.4.9.1 Описание

Данная функция получает биометрические данные от присоединенного устройства (модуля датчика) и сравнивает их с набором контрольных ЗБИ *(Population)*.

Совокупность, с которой проводится сопоставление, может быть представлена одним из двух способов.

a) в базе данных ЗБИ, обозначаемой открытым дескриптором базы данных;

b) во входном параметре в виде массива ЗБИ.

Примечание — При использовании управляемой БПУ базы данных ЗБИ она должна быть сначала открыта путем вызова функции ***BioAPI_DbOpen.***

Существует возможность использования массива ЗБИ, который может быть определен в BioAPI_IDENTIFY_POPULATION_TYPE в структуре BioAPI_IDENTIFY_POPULATION. Если он определен как BioAPI_PRESET_ARRAY_TYPE (3), то будет использоваться массив ЗБИ, который был предварительно установлен при вызове функции ***BioAPI_PresetIdentifyPopulation.*** Предварительно установленный массив ЗБИ будет освобожден ПБУ внутренне при вызове ***BioAPI_BSPDetach.***

Приложение должно запросить максимальное значение критерия ОЛС успешного сопоставления.

Функция выполняет действия в следующем порядке:

a) получает образец и обрабатывает его соответствующим образом;

b) определяет число кандидатов популяции, чья оценка схожести соответствует определенному критерию;

c) выделяет область памяти, достаточную для размещения массива элементов типа BioAPI_CANDIDATE с числом, равным числу кандидатов, определенных в перечислении b);

d) заполняет массив информацией о кандидатах, определенных в перечислении b), включая *FMRAchieved* каждого кандидата;

e) возвращает адрес массива в параметре *Candidates* и размер массива в параметре *NumberOfResults*.

Примечание — Дополнительная информация относительно использования понятия ОЛС для нормализованного оценивания схожести и выбора порога приведена в приложении С, раздел С.4.

По умолчанию за предоставление связанного с операцией идентификации интерфейса пользователя отвечает ПБУ. Приложение может запросить управление видом и поведением ГИП в виде предоставления указателя обратного вызова ГИП в функции ***BioAPI_SetGUICallbacks.*** Дополнительное описание особенностей интерфейса пользователя приведено в приложении С, раздел С.7.

Так как функция ***BioAPI_Identify*** включает в себя функцию получения данных, она упорядочивает использование устройств получения биометрических данных. Если два или более биометрических приложения одновременно запрашивают устройства регистрации биометрических данных, «опоздавшие» должны будут ждать завершения операции или истечения времени ожидания. Сериализацию осуществляет ПБУ. Это же может быть реализовано либо возвращением ошибки «занято» (BioAPI_UNIT_IN_USE), либо организацией очередности.

Область памяти, возвращаемая вызовом функции БиоАПИ, должна быть освобождена приложением с помощью вызова функции ***BioAPI_Free*** в том случае, если приложение больше не будет ее использовать (см. 8.7.2). Выходные ЗБИ могут быть восстановлены путем вызова функции ***BioAPI_GetBIRFromHandle,*** которая освобождает дескриптор, или дескриптор может быть освобожден без восстановления ЗБИ путем вызова функции ***BioAPI_FreeBIRHandle.***

8.4.9.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

MaxFMRRequested (входной) — значение ОЛС, являющееся критерием успешной идентификации (порог сопоставления).

Subtype (входной/необязательный) — определяет подтип регистрации (например, левый или правый глаз). Значение BioAPI_NO_SUBTYPE_AVAILABLE (0x00) указывает, что подтип должен выбрать ПБУ.

Примечание — Не все ПБУ поддерживают получение данных определенного подтипа.

Population (входной) — совокупность контрольных ЗБИ (шаблонов), по которой проводится идентификация (с помощью ПБУ).

TotalNumberOfTemplates (входной) — определяет общее число шаблонов, сохраненной приложением совокупности. Нулевое значение указывает, что приложение не предоставляет число шаблонов.

Примечание — Если совокупность распределена по нескольким базам данных / частям, то общий размер совокупности будет больше совокупности, которую видит ПБУ. ПБУ может отображать *FARRequested* на свой внутренний порог сопоставления, основываясь на этом общем размере совокупности.

Binning (входной) — Булево значение, указывающее, включен или нет режим категоризации.

Примечание 1 — Категоризация — методика оптимизации поиска, которую может использовать ПБУ. Данная методика основана на поиске подмножества совокупности по внутренним характеристикам биометрических данных. Наряду с увеличением скорости операции сопоставления, может также увеличиться вероятность пропуска кандидата из-за возможности ошибочной буферизации и в результате буфер не будет содержать соответствующую ЗБИ.

Примечание 2 — Дополнительная информация относительно категоризации приведена в приложении А, подпункт А.4.6.2.10.

MaxNumberOfResults (входной) — определяет максимальное число кандидатов, возвращаемых в результате сопоставления 1:N. Нулевое значение указывает на необходимость возврата всех кандидатов.

NumberOfResults (выходной) — указатель на число кандидатов, возвращенных в массиве Candidates в качестве результата сопоставления 1:N.

Candidates (выходной) — указатель на адрес массива элементов типа BioAPI_CANDIDATE, содержащих информацию о ЗБИ, идентифицированных в результате процесса сопоставления (то есть индексы, связанные с ЗБИ, для которых был превышен порог сопоставления). Данный перечень является ранжированным, причем запись с самой высокой оценкой схожести находится на первом месте. Если ни одного кандидата не найдено, данный указатель будет пустым. Если *Population* была представлена в базе данных ЗБИ, т. е. значение BioAPI_IDENTIFY_POPULATION_TYPE равно BioAPI_DB_TYPE, то массив *Candidates* содержит указатели на УУИД, связанные с ЗБИ, хранящимися во внутренней базе данных регистрации ПБУ. Если *Population* была представлена как передаваемый массив ЗБИ, то BioAPI_IDENTIFY_POPULATION_TYPE имеет значение BioAPI_ARRAY_TYPE и массив *Candidates* содержит указатели на относительные индексы в передаваемом массиве.

Timeout (входной) — целое число, определяющее значение времени ожидания (в миллисекундах) для выполнения операции. Если время ожидания истекло, функция возвращает ошибку, память для массива не выделяется и возвращается пустой указатель в параметре *Candidates*. Данное значение может быть

любым положительным числом. Значение минус 1 означает, что будет использоваться значение времени ожидания, заданное ПБУ по умолчанию.

AuditData (выходной/необязательный) — дескриптор ЗБИ, содержащей исходные биометрические данные. Эти данные могут использоваться для предоставления личных биометрических данных. Нулевой указатель на входе указывает, что контрольные данные не должны быть собраны. Не все ПБУ поддерживают сбор контрольных данных. ПБУ может возвратить значение дескриптора BioAPI_UNSUPPORTED_BIR_HANDLE, чтобы указать, что *AuditData* не поддерживается, или значение BioAPI_INVALID_BIR_HANDLE, чтобы указать, что контрольные данные недоступны.

8.4.9.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.4.9.4 Ошибки

BioAPIERR_USER_CANCELLED
BioAPIERR_UNABLE_TO_CAPTURE
BioAPIERR_TOO_MANY_HANDLES
BioAPIERR_BIR_SIGNATURE_FAILURE
BioAPIERR_TIMEOUT_EXPIRED
BioAPIERR_NO_INPUT_BIRS
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_INCONSISTENT_PURPOSE
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_QUALITY_ERROR
BioAPIERR_UNIT_IN_USE
BioAPIERR_PRESET_BIR_DOES_NOT_EXIST
BioAPIERR_INVALID_DB_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

Примечание 1 — Не все ПБУ поддерживают идентификацию1:N. Данное указание должно быть приведено в руководстве по программированию конкретного ПБУ.

Примечание 2 — В зависимости от ПБУ, местоположения и размера базы данных, по которой производится поиск, выполнение данной операции может занять значительное время. Рекомендуемое значение параметра *Timeout* должно быть указано в руководстве по программированию конкретного ПБУ.

Примечание 3 — Число кандидатов сопоставления, найденных ПБУ, зависит от текущего значения ОЛД алгоритма сопоставления, которое указано в параметре установок порога *MaxFMRRequested*.

8.4.10 Функция BioAPI_Import

```
BioAPI_RETURN BioAPI BioAPI_Import
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_DATA *InputData,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *InputFormat,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
  BioAPI_BIR_PURPOSE Purpose,
  BioAPI_BIR_HANDLE *ConstructedBIR);
```

8.4.10.1 Описание

Данная функция передает исходные биометрические данные, полученные биометрическим приложением любым способом, и запрашивает указанный вызов ПБУ о необходимости создания ЗБИ определенного назначения. Параметр *InputData* определяет буфер памяти, содержащий исходные биометрические данные, а *InputFormat* определяет формат исходных биометрических данных. Форматы *InputFormats*, которые может принять конкретный ПБУ, определяются самим ПБУ (см. ошибку BioAPIERR_UNSUPPORTED_FORMAT). Функция возвращает дескриптор к *ConstructedBIR*. Если приложению необходимо получить ЗБИ для их сохранения в базе данных или пересылки на сервер, оно может извлечь их путем вызова функции ***BioAPI_GetBIRFromHandle*** или сохранить путем вызова функции ***BioAPI_DbStoreBIR.***

Выходные *ConstructedBIR* могут быть восстановлены путем вызова функции ***BioAPI_GetBIRFromHandle,*** которая освобождает дескриптор, или дескриптор может быть освобожден без восстановления ЗБИ путем вызова функции ***BioAPI_FreeBIRHandle.***

8.4.10.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

InputData (входной) — указатель на данные изображения/потоковые данные для импорта в обработанную ЗБИ. Изображение/поток соответствует формату, указанному в *InputFormat*.

InputFormat (входной) — формат *InputData*.

OutputFormat (входной/необязательный) — определяет формат ББД возвращаемого *ConstructedBIR*, если ПБУ поддерживает более одного формата. Пустой указатель означает, что формат должен быть выбран ПБУ.

Purpose (входной) — параметр, указывающий назначение *ConstructedBIR*.

ConstructedBIR (выходной) — дескриптор ЗБИ, созданной из импортированных биометрических данных. Данная ЗБИ может быть промежуточной или обработанной ЗБИ (в соответствии с типом, указанным в заголовке).

8.4.10.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.4.10.4 Ошибки

BioAPIERR_UNSUPPORTED_FORMAT
BioAPIERR_UNABLE_TO_IMPORT
BioAPIERR_TOO_MANY_HANDLES
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_PURPOSE_NOT_SUPPORTED

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.4.11 Функция BioAPI_PresetIdentifyPopulation

```
BioAPI_RETURN BioAPI BioAPI_PresetIdentifyPopulation
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_IDENTIFY_POPULATION *Population);
```

8.4.11.1 Описание

Данная функция предоставляет совокупность ЗБИ ПБУ, указанному в BSPHandle. Совокупность шаблонов, с которыми проводится сопоставление, представлена одним из следующих способов:

a) в базе данных ЗБИ, обозначаемой открытым дескриптором базы данных;

b) во входном параметре в виде массива ЗБИ;

ПБУ выделяет область памяти и передает в нее совокупность ЗБИ в формате, который поддерживается внутренним алгоритмом ПБУ (не стандартизован). После успешного вызова данной функции приложение может вызвать функцию ***BioAPI_Identify*** или ***BioAPI_IdentifyMatch,*** задавая BioAPI_PRESET_ARRAY_TYPE в структуре BioAPI_IDENTIFY_POPULATION. ПБУ сохраняет этот блок памяти до тех пор, пока снова не будет вызвана функция ***BioAPI_PresetIdentifyPopulation*** или ***BioAPI_BSPDetach.***

8.4.11.2 Параметры

BSPHandle (входной) — дескриптор присоединенного поставщика услуги БиоАПИ.

Population (входной) — совокупность ЗБИ, по которой проводится идентификация.

8.4.11.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.4.11.4 Ошибки

BioAPIERR_INVALID_BIR_HANDLE
BioAPIERR_BIR_SIGNATURE_FAILURE
BioAPIERR_NO_INPUT_BIRS
BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_INCONSISTENT_PURPOSE
BioAPIERR_BIR_NOT_FULLY_PROCESSED
BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_QUALITY_ERROR
BioAPIERR_FUNCTION_FAILED

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

Примечание — В зависимости от размера передаваемой базы данных и требуемой степени преобразования выполнение данной операции может занять значительное время.

8.5 Операции над базой данных

Операции над базой данных обеспечивают доступ приложения к базам данных ЗБИ архива модулей БиоАПИ (либо непосредственно управляемыми ПБУ, либо реализованными через ПБФ).

Примечание 1 — Описание контекста и использования функций базы данных приведено в приложении С, раздел С.6.

Примечание 2 — Механизмы авторизации для управления правом биометрического приложения создавать, удалять или производить запись в базу данных в настоящем стандарте не рассматриваются (но могут ссылаться на ИСО/МЭК 19784-2 «Информационные технологии — Программный интерфейс биометрического приложения — Часть 2: Интерфейс поставщика функции биометрического архива»).

8.5.1 Функция BioAPI_DbOpen

```
BioAPI_RETURN BioAPI BioAPI_DbOpen
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_UUID *DbUuid,
  BioAPI_DB_ACCESS_TYPE AccessRequest,
  BioAPI_DB_HANDLE *DbHandle,
  BioAPI_DB_MARKER_HANDLE *MarkerHandle);
```

8.5.1.1 Описание

Данная функция открывает базу данных ЗБИ, поддерживаемую текущим присоединенным архивом установленного вызова ПБУ, используя режим доступа, указанный в *AccessRequest*. Создается новый маркер и устанавливается на первую запись в базе данных ЗБИ, а затем возвращается дескриптор данного маркера.

Некоторые ПБУ могут поддерживать только отдельную базу данных ЗБИ или иметь предпочтительную базу данных. Приложение может предоставить ПБУ возможность выбрать базу данных ЗБИ для открытия, используя значение пустого указателя для параметра УУИД базы данных.

Примечание — Каждый вызов данной функции должен сопровождаться вызовом функции ***BioAPI_DbFreeMarker*** для освобождения ресурсов маркера. В приложении должно быть учтено, что каждый вызов функции ***BioAPI_DbOpen*** приводит к выделению ресурсов, которые не могут быть автоматически освобождены, в противном случае будет невозможно выполнять итерации в базе данных с использованием возвращенного дескриптора маркера. Если приложение не требует выполнения итераций в базе данных, то оно должно сразу же вызвать функцию ***BioAPI _DbFreeMarker,*** в противном случае оно должно выполнить это после завершения итерации.

8.5.1.2 Параметры

BSPHandle (входной) — дескриптор присоединенного вызова ПБУ.

DbUuid (входной) — указатель на УУИД, определяющий открываемую базу данных ЗБИ. Если указатель пустой, то ПБУ должен выбрать базу данных ЗБИ.

AccessRequest (входной) — признак требуемого режима доступа к базе данных ЗБИ BioAPI_DB_ACCESS_READ или BioAPI_DB_ACCESS_WRITE. В отдельной системе в каждый момент времени только одно приложение может открыть адресуемую базу данных в режиме записи BioAPI_DB_ACCESS_WRITE.

DbHandle (выходной) — дескриптор открытой базы данных ЗБИ. Если выполнение функции оканчивается ошибкой, данное значение будет установлено в BioAPI_DB_INVALID_HANDLE.

Marker (выходной) — дескриптор признака, который впоследствии может использоваться приложением для выполнения итераций в базе данных ЗБИ, начиная с первой записи.

8.5.1.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.1.4 Ошибки

BioAPIERR_UNABLE_TO_OPEN_DATABASE
BioAPIERR_DATABASE_IS_OPEN
BioAPIERR_DATABASE_IS_LOCKED
BioAPIERR_DATABASE_DOES_NOT_EXIST

BioAPIERR_INVALID_UUID
BioAPIERR_INVALID_ACCESS_REQUEST
BioAPIERR_DATABASE_IS_CORRUPT
Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.2 Функция BioAPI_DbClose

```
BioAPI_RETURN BioAPI BioAPI_DbClose
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle);
```

8.5.2.1 Описание

Данная функция закрывает открытую базу данных ЗБИ. Все маркеры, установленные для записей в базе данных, освобождаются и их дескрипторы становятся недействительны.

Примечание — Если база данных, открытая в режиме BioAPI_DB_ACCESS_WRITE не будет закрыта, она может быть повреждена.

8.5.2.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

DbHandle (входной) — дескриптор БД для открытой базы данных ЗБИ, управляемой ПБУ. Данный параметр определяет открытую базу данных, которая должна быть закрыта.

8.5.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.2.4 Ошибки

BioAPIERR_UNABLE_TO_CLOSE_DATABASE
BioAPIERR_INVALID_DB_HANDLE
BioAPIERR_DATABASE_IS_CORRUPT
Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.3 Функция BioAPI_DbCreate

```
BioAPI_RETURN BioAPI BioAPI_DbCreate
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_UUID *DbUuid,
  uint32_t NumberOfRecords,
  BioAPI_DB_ACCESS_TYPE AccessRequest,
  BioAPI_DB_HANDLE *DbHandle);
```

8.5.3.1 Описание

Данная функция создает и открывает новую базу данных ЗБИ для присоединенного в данный момент модуля архива установленного вызова ПБУ. Идентификация новой базы данных определяется входным параметром *DbUuid,* который должен быть создан биометрическим приложением и должен отличаться от УУИД всех баз данных, как открытых, так и закрытых в данный момент, поддерживаемых данным модулем архива. Вновь созданная база данных ЗБИ может быть открыта в указанном режиме доступа.

Примечание — Функция используется для создания новой базы данных ЗБИ. Она не передает никакой информации модулю архива о новой базе данных, за исключением ее УУИД и условий доступа. Существуют архивы, которые могут работать только с базами данных статического размера или которым требуются значительные усилия для управления базой данных с динамически изменяемым размером (например, смарт-картами, которые сохраняют шаблоны в «прозрачных» или структурированных файлах и имеют статический размер, зависящий от характеристик операционной системы смарт-карты). Для создания новой базы данных ЗБИ архиву может потребоваться информация о ее размере. Так как вызывающее приложение может не знать средний или максимальный размер шаблона (в байтах), то оно предоставляет максимальное число записей, которые будут сохранены в базе данных. Если архив может работать с динамическими базами данных, то он будет игнорировать параметр *NumberOfRecords.*

8.5.3.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

DbUuid (входной) — указатель на УУИД, который будет идентифицировать создаваемую базу данных ЗБИ.

NbrRecords (входной) — максимальное число записей, которые будут сохранены в базе данных ЗБИ.

AccessRequest (входной) — признак требуемого режима доступа к базе данных ЗБИ: «чтение» или «запись». В конкретной системе в каждый момент времени только одно приложение может открыть адресуемую базу данных в режиме записи BioAPI_DB_ACCESS_WRITE.

DbHandle (выходной) — дескриптор вновь созданной и открытой базы данных. Если выполнение функции оканчивается ошибкой, данное значение будет установлено в BioAPI_DB_INVALID_HANDLE.

8.5.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.3.4 Ошибки

BioAPIERR_UNABLE_TO_CREATE_DATABASE
BioAPIERR_DATABASE_ALREADY_EXISTS
BioAPIERR_INVALID_UUID
BioAPIERR_INVALID_ACCESS_REQUEST
Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.4 Функция BioAPI_DbDelete

```
BioAPI_RETURN BioAPI BioAPI_DbDelete
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_UUID *DbUuid);
```

8.5.4.1 Описание

Данная функция удаляет все записи из указанной базы данных ЗБИ, а также всю информацию о состоянии, связанную с этой базой данных.

8.5.4.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

DbUuid (входной) — указатель на УУИД, идентифицирующий удаляемую базу данных ЗБИ.

8.5.4.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.4.4 Ошибки

BioAPIERR_DATABASE_IS_OPEN
BioAPIERR_INVALID_UUID
Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.5 Функция BioAPI_DbSetMarker

```
BioAPI_RETURN BioAPI BioAPI_DbSetMarker
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle,
  const BioAPI_UUID *KeyValue,
  BioAPI_DB_MARKER_HANDLE MarkerHandle);
```

8.5.5.1 Описание

Данная функция устанавливает маркер, идентифицированный параметром MarkerHandle, для указания на запись, обозначенную параметром *KeyValue* в базе данных ЗБИ, указанной в *DbHandle*. Пустой указатель приведет к установке маркера на первую запись в базе данных.

Примечание — При возникновении ошибки положение маркера не изменяется.

8.5.5.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

DbHandle (входной) — дескриптор открытой базы данных ЗБИ присоединенного архива модулей БиоАПИ.

KeyValue (входной) — ключ в базе данных ЗБИ, на который должен быть установлен маркер.

MarkerHanle (входной/выходной) — дескриптор маркера, который должен быть установлен и может быть впоследствии использован приложением для выполнения итераций в базе данных ЗБИ начиная с новой позиции.

8.5.5.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.5.4 Ошибки

BioAPIERR_INVALID_DB_HANDLE

BioAPIERR_RECORD_NOT_FOUND

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.6 Функция BioAPI_DbFreeMarker

```
BioAPI_RETURN BioAPI BioAPI_DbFreeMarker
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_MARKER_HANDLE MarkerHandle);
```

8.5.6.1 Описание

Данная функция освобождает память и ресурсы, связанные с указанным маркером, и аннулирует MarkerHandle.

8.5.6.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

MarkerHandle (входной) — дескриптор маркера базы данных ЗБИ, который должен быть освобожден.

8.5.6.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.6.4 Ошибки

BioAPIERR_MARKER_HANDLE_IS_INVALID

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.7 Функция BioAPI_DbStoreBIR

```
BioAPI_RETURN BioAPI BioAPI_DbStoreBIR
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_INPUT_BIR *BIRToStore,
  BioAPI_DB_HANDLE DbHandle,
  BioAPI_UUID *BirUuid);
```

8.5.7.1 Описание

Данная функция сохраняет ЗБИ, идентифицированную параметром *BIRToStore*, в открытой базе данных ЗБИ, идентифицированной параметром *DbHandle*. Если *BIRToStore* определен дескриптором ЗБИ, то входной дескриптор BIR освобождается. Если *BIRToStore* определен значением ключа базы данных, то ЗБИ восстанавливается и сохраняется (копируется) в открытую базу данных. Новой ЗБИ в базе данных присваивается новый УУИД, который может использоваться в качестве ключевого значения для последующего доступа к ЗБИ.

8.5.7.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

BIRToStore (входной) — ЗБИ, сохраняемая в открытой базе данных (или ЗБИ, ее дескриптор или индекс к ней, сохраняемые в другой открытой базе данных).

DbHandle (входной) — дескриптор открытой базы данных ЗБИ.

Uuid (выходной) — УУИД, однозначно идентифицирующий новую ЗБИ в базе данных. Данный УУИД не может быть изменен. Для того чтобы связать с пользователем другую ЗБИ, необходимо удалить старую, сохранить в базе данных новую ЗБИ, а затем заменить старый УУИД новым в учетной записи базы данных приложения. ЗБИ должна быть добавлена в конце базы данных.

8.5.7.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.7.4 Ошибки

BioAPIERR_INVALID_DB_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.8 Функция BioAPI_DbGetBIR

```
BioAPI_RETURN BioAPI BioAPI_DbGetBIR
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle,
  const BioAPI_UUID *KeyValue,
  BioAPI_BIR_HANDLE *RetrievedBIR,
  BioAPI_DB_MARKER_HANDLE *MarkerHandle);
```

8.5.8.1 Описание

Данная функция извлекает ЗБИ, идентифицированную параметром *KeyValue* в открытой базе данных ЗБИ, идентифицированной параметром *DbHandle*. Функция копирует ЗБИ в память ПБУ и возвращает ее дескриптор. Маркер создается и устанавливается на запись, следующую за восстановленной ЗБИ в базе данных (или на первую запись базы данных, если восстановленная ЗБИ является последней), и возвращается дескриптор на маркер.

Область памяти, возвращаемая при вызове функции БиоАПИ, должна быть освобождена приложением путем вызова функции ***BioAPI_Free*** (8.7.2).

8.5.8.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

DbHandle (входной) — дескриптор открытой базы данных ЗБИ.

KeyValue (входной) — ключ в базе данных извлекаемой ЗБИ.

RetrievedBIR (выходной) — дескриптор извлеченной ЗБИ.

MarkerHandle (выходной) — дескриптор маркера, который может быть впоследствии использован приложением для выполнения итераций в базе данных ЗБИ, начиная с позиции восстановленной ЗБИ.

8.5.8.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.8.4 Ошибки

BioAPIERR_RECORD_NOT_FOUND

BioAPIERR_INVALID_DB_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.9 Функция BioAPI_DbGetNextBIR

```
BioAPI_RETURN BioAPI BioAPI_DbGetNextBIR
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle,
  BioAPI_DB_MARKER_HANDLE MarkerHandle,
  BioAPI_BIR_HANDLE *RetrievedBIR,
  BioAPI_UUID *BirUuid);
```

8.5.9.1 Описание

Данная функция извлекает ЗБИ, идентифицированную параметром *MarkerHandle*. Функция копирует ЗБИ в память ПБУ и возвращает ее дескриптор, а также указатель на УУИД, который однозначно идентифицирует ЗБИ в базе данных, после чего маркер устанавливается на следующую запись в базе данных.

Примечание — Если в базе данных больше нет записей, маркер будет указывать на несуществующую позицию.

8.5.9.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

DbHandle (входной) — дескриптор открытой базы данных ЗБИ.

MarkerHandle (входной/выходной) — дескриптор маркера, указывающий, какая запись должна быть восстановлена.

RetrievedBIR (выходной) — дескриптор извлеченной ЗБИ.

BirUuid (выходной) — УУИД, однозначно идентифицирующий извлеченную ЗБИ в базе данных.

8.5.9.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.9.4 Ошибки

BioAPIERR_END_OF_DATABASE

BioAPIERR_MARKER_HANDLE_IS_INVALID

BioAPIERR_INVALID_DB_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.5.10 Функция BioAPI_DbDeleteBIR

```
BioAPI_RETURN BioAPI BioAPI_DbDeleteBIR
    (BioAPI_HANDLE BSPHandle,
    BioAPI_DB_HANDLE DbHandle,
    const BioAPI_UUID *KeyValue);
```

8.5.10.1 Описание

ЗБИ, идентифицированная параметром *KeyValue* в открытой базе данных ЗБИ, идентифицированной параметром *DbHandle*, удаляется из базы данных. Если маркер установлен на удаленную запись, то:

a) если эта ЗБИ была не последней записью в базе данных, то маркер перемещается на следующую ЗБИ;

b) в другом случае маркер устанавливается на недействительную позицию. Однако дескриптор маркера остается действительным и может быть использован при последующем вызове функции ***BioAPI_DbSetMarker*** для установки маркера в позицию другой записи.

8.5.10.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

DbHandle (входной) — дескриптор открытой базы данных ЗБИ.

KeyValue (входной) — УУИД ЗБИ, которая должна быть удалена.

8.5.10.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.5.10.4 Ошибки

BioAPIERR_RECORD_NOT_FOUND
BioAPIERR_INVALID_DB_HANDLE
Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.6 Операции с модулями БиоАПИ

8.6.1 Функция BioAPI_SetPowerMode

BioAPI_RETURN BioAPI BioAPI_SetPowerMode
(BioAPI_HANDLE BSPHandle,
BioAPI_UNIT_ID UnitId,
BioAPI_POWER_MODE PowerMode).

8.6.1.1 Описание

Данная функция устанавливает присоединенный модуль БиоАПИ соответствующей прикрепленной сессии ПБУ в требуемый режим энергопотребления в том случае, если модуль поддерживает этот режим.

8.6.1.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

UnitId (входной) — ИД модуля БиоАПИ, для которого должен быть установлен режим энергопотребления. Опция BioAPI_DONT_CARE недоступна для данной функции.

PowerMode (входной) — 32-битовое значение, указывающее режим энергопотребления, в который требуется перевести модуль БиоАПИ.

8.6.1.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.6.1.4 Ошибки

BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_NOT_INSERTED
BioAPIERR_UNIT_IN_USE
Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.6.2 Функция BioAPI_SetIndicatorStatus

BioAPI_RETURN BioAPI BioAPI_SetIndicatorStatus
(BioAPI_HANDLE BSPHandle,
BioAPI_UNIT_ID UnitId,
BioAPI_INDICATOR_STATUS IndicatorStatus).

8.6.2.1 Описание

Данная функция устанавливает выбранный модуль в требуемый режим индикации в том случае, если модуль поддерживает этот режим. После того как параметру *IndicatorStatus* присвоено значение BioAPI_INDICATOR_ACCEPT или BioAPI_INDICATOR_REJECT, режим индикации остается неизменным до тех пор, пока приложение не установит другое значение.

8.6.2.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

UnitId (входной) — ИД модуля БиоАПИ, для которого должен быть установлен режим индикации. Опция BioAPI_DONT_CARE недоступна для данной функции.

IndicatorStatus (входной) — значение устанавливаемого режима индикации модуля БиоАПИ.

8.6.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.6.2.4 Ошибки

BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_NOT_INSERTED
BioAPIERR_UNIT_IN_USE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.6.3 Функция BioAPI_GetIndicatorStatus

```
BioAPI_RETURN BioAPI BioAPI_GetIndicatorStatus
  (BioAPI_HANDLE BSPHandle,
  BioAPI_UNIT_ID UnitId,
  BioAPI_INDICATOR_STATUS *IndicatorStatus);
```

8.6.3.1 Описание

Данная функция возвращает режим индикации модуля БиоАПИ в том случае, если модуль поддерживает этот режим.

8.6.3.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

UnitId (входной) — ИД модуля БиоАПИ, для которого должен быть получен режим индикации. Опция BioAPI_DONT_CARE недоступна для данной функции.

IndicatorStatus (выходной) — значение режима индикации модуля БиоАПИ.

8.6.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.6.3.4 Ошибки

BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_NOT_INSERTED
BioAPIERR_UNIT_IN_USE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.6.4 Функция BioAPI_CalibrateSensor

```
BioAPI_RETURN BioAPI BioAPI_CalibrateSensor
  (BioAPI_HANDLE BSPHandle,
  int32_t Timeout);
```

8.6.4.1 Описание

Данная функция выполняет калибровку подключенного датчика модуля БиоАПИ в том случае, если модуль датчика поддерживает калибровку.

8.6.4.2 Параметры

BSPHandle (входной) — дескриптор присоединенного поставщика биометрической услуги.

Timeout (входной) — целое число, определяющее значение времени ожидания (в миллисекундах) для операции. Если время ожидания истекло, функция возвращает ошибку. Данное значение может быть любым числом. Значение минус 1 означает, что ПБУ должен использовать значение времени ожидания калибровки по умолчанию.

8.6.4.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.6.4.4 Ошибки

BioAPIERR_FUNCTION_NOT_SUPPORTED
BioAPIERR_UNIT_IN_USE
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_NOT_INSERTED
BioAPIERR_CALIBRATION_NOT_SUCCESSFUL
BioAPIERR_TIMEOUT_EXPIRED

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.7 Служебные операции

8.7.1 Функция BioAPI_Cancel

```
BioAPI_RETURN BioAPI BioAPI_Cancel
  (BioAPI_HANDLE BSPHandle);
```

8.7.1.1 Описание

Данная функция должна отменить все заблокированные в данный момент вызовы, связанные с *BSPHandle*. Данная функция не должна завершаться до тех пор, пока все заблокированные вызовы не будут отменены.

8.7.1.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

8.7.1.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.7.1.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.7.2 Функция BioAPI_Free

```
BioAPI_RETURN BioAPI BioAPI_Free
  (void* Ptr);
```

8.7.2.1 Описание

Данная функция освобождает область памяти, на которую указывает параметр *Ptr*. Если *Ptr* имеет нулевое значение, никакие действия не выполняются. В противном случае, если *Ptr* не соответствует указателю, ранее возвращенному функциями БиоАПИ, или если область памяти уже была освобождена вызовом функции ***BioAPI_Free,*** последствия вызова данной функции неопределенны.

Также используют функции BioSPI, в которых ПБУ выделяет область памяти, которая должна быть освобождена инфраструктурой путем вызова функции ***BioSPI_Free.*** Каждый раз, когда инфраструктура посылает блок памяти приложению, оно становится ответственным за освобождение данного блока памяти. В таких случаях приложение вызывает функцию ***BioAPI_Free*** и инфраструктура (или получатель вызова) должна или освободить блок памяти, или вызвать функцию ***BioSPI_Free*** прикрепленной сессии соответствующего ПБУ.

Примечание — Инфраструктура может быть реализована таким образом, что ПБУ распределяет указатели, которые не передаются приложению. В таком случае инфраструктура перемещает данные, возвращенные ПБУ, в выделенную область памяти и после копирования в область памяти до возвращения в приложение вызывает функцию ***BioAPI_Free.*** Однако такие действия инфраструктуры не сообщаются приложению.

В других случаях инфраструктура сама распределяет память и передает приложению указатель на нее. В таких случаях, когда приложение вызывает ***BioAPI_Free,*** инфраструктура должна освободить область памяти, не вызывая функцию ***BioSPI_Free*** ПБУ.

Необходимо, чтобы инфраструктура сохраняла информацию о том, какие блоки памяти она распределила сама и какие блоки памяти были получены от ПБУ. В дальнейшем она должна хранить информацию о ПБУ и прикрепленных сессиях, в которых инфраструктура получила блок памяти.

8.7.2.2 Параметры

Ptr (входной) — указатель на освобождаемую память.

8.7.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.7.2.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

9 Интерфейс поставщика услуги БиоАПИ

9.1 Общие положения

ИПУ — программный интерфейс, который предоставляет ПБУ для подключения к инфраструктуре БиоАПИ. ИПУ является взаимно однозначным отображением вызова функции от биометрического приложения к инфраструктуре БиоАПИ (использующей ПИП, указанный в разделе 8) и далее — к присоединенной сессии. Инфраструктура БиоАПИ направляет вызовы ПИП к соответствующему ИПУ указанного ПБУ (определяемого параметром дескриптора ПБУ в функции ПИП). В данном разделе были приняты следующие соглашения:

a) для функции ИПУ, параметры которой идентичны параметрам функции ПИП, имеющей то же самое наименование (кроме пропуска дескриптора ПБУ), в данном разделе приведена только ее сигнатура, но приведено подробное описание соответствующей функции ПИП;

b) для функции ИПУ, параметры которой отличаются от параметров функции ПИП, имеющей то же самое наименование, приведено ее полное описание;

c) существуют функции ПИП, не имеющие соответствующих функций ИПУ; такие функции полностью не обрабатываются инфраструктурой БиоАПИ.

9.2 Определение типов (для ПБУ)

9.2.1 BioSPI_EventHandler

Прототип функции ***BioAPI_EventHandler*** (см. 7.28) используется для определения интерфейса обработчика событий, которые позволяют инфраструктуре БиоАПИ принимать от ПБУ асинхронные уведомления о событиях типа BioAPI_EVENT.

Пример — события вставки или удаления модуля БиоАПИ, обнаружения неисправности.

Адрес функции ***BioSPI_EventHandler*** передается ПБУ во время выполнения функции ***BioSPI_BSPLoad.*** Данный обработчик событий является единственным, который должен вызывать ПБУ для уведомления инфраструктуры БиоАПИ о типах событий, происходящих в загруженном ПБУ.

Для каждого ПБУ, загруженного одним или несколькими приложениями в результате выполнения функции ***BioAPI_BSPLoad,*** существует единственный обработчик события. Инфраструктура БиоАПИ должна уведомлять обработчик о событиях биометрических приложений, загрузивших ПБУ, в котором происходит событие. Если инфраструктура получила уведомление о событии подключения до того, как данное биометрическое приложение загрузило ПБУ (но после того, как другое биометрическое приложение загрузило его), инфраструктура должна запомнить данное событие и уведомить о нем обработчик событий биометрического приложения немедленно после загрузки ПБУ биометрическим приложением.

Инфраструктура БиоАПИ пересылает события биометрическому приложению, которое вызвало соответствующую функцию ***BioAPI_BSPLoad.*** Обработчик, указанный в *BioSPI_EventHandler*, может быть вызван многократно в ответ на одно событие.

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

9.2.1.1 Определения

BSPUuid — УУИД ПБУ, вызывающего событие.

UnitID — ИД модуля БиоАПИ, связанного с данным событием.

UnitSchema — указатель на модуль схемы модуля БиоАПИ, связанного с данным событием.

EventType — произошедшее событие типа BioAPI_EVENT.

Если параметр *EventType* имеет значение BioAPI_NOTIFY_INSERT, то должна быть подготовлена схема модуля (то есть параметр *UnitSchema* должен указывать на переменную типа). В другом случае параметр *UnitSchema* должен быть пустым.

Когда инфраструктура получает от ПБУ вызов обработчика событий, который хранит схему модуля, инфраструктура не должна вызывать функцию ***BioSPI_Free*** для освобождения блока памяти, содержащего схему модуля или блок памяти, указывающий на что-либо любым своим элементом.

9.2.2 BioSPI_BFP_ENUMERATION_HANDLER

Данная функция представляет собой обратный вызов, с помощью которого инфраструктура БиоАПИ открывает ПБУ, чтобы дать ему возможность получать информацию об установленных ПБУ. Данная функция аналогична функции ***BioAPI_EnumBFPs*** (см. 8.1.10), но в отличие от нее не является частью ПИП

БиоАПИ и поэтому не открыта приложению. И наоборот, ПБУ может использовать данную функцию для получения той же информации, которую приложение может получить путем вызова функции ***BioAPI_EnumBFPs.***

Адрес обратного вызова BioSPI_BFP_ENUMERATION_HANDLER инфраструктура предоставляет ПБУ в виде входного параметра функции ***BioSPI_BSPLoad.***

9.2.2.1 Определения

```
typedef BioAPI_RETURN (*BioSPI_BFP_ENUMERATION_HANDLER)
  (BioAPI_BFP_SCHEMA **BFPSchemaArray,
  uint32_t *NumberOfElements);
```

Функция обратного вызова предоставляет информацию о всех ПБФ, установленных в реестре компонентов. Функция выполняет действия в следующем порядке:

a) выделяет область памяти, достаточную для размещения массива элементов типа BioAPI_BFP_SCHEMA с числом элементов, равным числу установленных ПБФ;

b) заполняет массив схемами всех установленных ПБФ;

c) возвращает адрес массива в параметре BFPSchemaArray и число элементов массива в параметре NumberOfElements.

Область памяти, содержащая массив, должна быть освобождена с помощью ПБУ путем обратного вызова обработчика освобождения памяти инфраструктуры (см. 9.2.3) в том случае, если данный блок памяти в дальнейшем не будет использоваться ПБУ.

Область памяти, на которой указывают параметры Path и BFPProperty, в пределах каждого элемента массива, должна быть также освобождена с помощью ПБУ путем обратного вызова обработчика освобождения памяти инфраструктуры (см. 9.2.3), если приложение в дальнейшем не будет их использовать.

9.2.2.2 Параметры

BFPSchemaArray (выходной) — указатель на адрес массива элементов типа BioAPI_BFP_SCHEMA (распределенного инфраструктурой), содержащего информацию о схемах ПБФ.

NumberOfElements (выходной) — указатель на число элементов массива или число схем ПБФ в реестре компонентов.

9.2.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

9.2.2.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

9.2.3 BioSPI_MEMORY_FREE_HANDLER

Данную функцию также называют обработчиком освобождения памяти. Она представляет собой обратный вызов, который инфраструктура БиоАПИ открывает для ПБУ, давая ему возможность вызвать функцию освобождения памяти, которая была выделена инфраструктурой для возвращения данных ПБУ во время приоритетного обратного вызова. Подобное распределение памяти происходит в тот момент, когда ПБУ вызывает обработчик перечислений ПБФ (см. 9.2.2). Функция освобождения памяти аналогична функции БиоАПИ ***BioAPI_Free*** (см. 8.7.2), но отличается от нее тем, что не является частью ПИП БиоАПИ и поэтому недоступна приложению.

Адрес обратного вызова BioSPI_MEMORY_FREE_HANDLER предоставляется ПБУ инфраструктурой как входной параметр функции ***BioSPI_BSPLoad.***

9.2.3.1 Определения

```
typedef BioAPI_RETURN (*BioSPI_MEMORY_FREE_HANDLER)
  (void* Ptr);
```

Данный обратный вызов производит освобождение той области памяти, на которую указывает параметр *Ptr*. Если *Ptr* пустой, никакого действия не происходит. Однако, если *Ptr* не соответствует указателю, ранее возвращенному с помощью обратного вызова, или если область памяти уже была освобождена ранее путем вызова обработчика освобождения памяти, поведение функции не определено.

Примечание — В отличие от функции ***BioAPI_Free*** никакие запросы обработчика освобождения памяти не приводят к вызову функции ***BioSPI_Free.***

9.2.3.2 Параметры

Ptr (входной) — указатель на область памяти, которая должна быть освобождена.

9.2.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

9.2.3.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

9.3 Биометрические операции поставщика услуги

9.3.1 Операции ИПУ управления компонентом

9.3.1.1 BioSPI_BSPLoad

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

9.3.1.1.1 Описание

Данная функция выполняет процесс инициализации компонента между БиоАПИ и ПБУ. Функция ***BioSPI_BSPLoad*** не должна вызываться более одного раза без соответствующего вызова функции ***BioSPI_BSPUnload.***

BSPUuid — идентифицирует вызываемый ПБУ.

BioAPINotifyCallback — определяет обратный вызов, используемый для уведомления инфраструктуры БиоАПИ о событиях типа BioAPI_EVENT в любой действующей прикрепленной сессии. ПБУ должен сохранять данную информацию для дальнейшего использования.

BFPEnumerationHandler — является адресом обратного вызова обработчика перечислений ПБФ, предоставляемым инфраструктурой для ПБУ. ПБУ должен сохранять адрес для дальнейшего использования. ПБУ может использовать данный обратный вызов каждый раз, когда необходимо получить информацию об установленных в инфраструктуре ПБФ.

MemoryFreeHandler — является адресом обратного вызова обработчика освобождения памяти, предоставляемым инфраструктурой для ПБУ. ПБУ должен сохранять адрес для дальнейшего использования. ПБУ может использовать данный обратный вызов каждый раз, когда необходимо освободить блок памяти, распределенный инфраструктурой во время приоритетного обратного вызова обработчика перечислений ПБФ.

Примечание — Данная функция аналогична функции ***BioAPI_BSPLoad*** (см. 8.1.5).

9.3.1.1.2 Параметры

BSPUuid (входной) — УУИД вызываемого ПБУ.

BioAPINotifyCallback (входной) — указатель на функцию для обработчика событий БиоАПИ, который управляет событиями типа BioAPI_EVENT.

BFPEnumerationHandler (входной) — функциональный указатель на обработчик перечислений ПБФ инфраструктуры, который возвращает информацию о схемах ПБФ для всех установленных ПБФ.

MemoryFreeHandler (входной) — указатель на функцию обработки освобождения памяти инфраструктуры.

9.3.1.1.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

9.3.1.1.4 Ошибки

Аналогичны указанным для функции ***BioAPI_BSPLoad*** (8.1.5.4).

9.3.1.2 BioSPI_BSPUnload

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

9.3.1.2.1 Описание

Данная функция блокирует события и выполняет дерегистрацию функции уведомления о событии. ПБУ может выполнить операции освобождения ресурсов, обратные процессу инициализации, выполненному при ***BioSPI_BSPLoad.***

Примечание — Данная функция аналогична функции ***BioAPI_BSPUnload*** (см. 8.1.6).

9.3.1.2.2 Параметры

BSPUuid (входной) — УУИД вызываемого ПБУ.

9.3.1.2.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

9.3.1.2.4 Ошибки

Аналогичны указанным для функции ***BioAPI_BSPUnload*** (8.1.6.4).

9.3.1.3 BioSPI_BSPAttach

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

9.3.1.3.1 Описание

Данная функция вызывается инфраструктурой один раз при каждом вызове функции ***BioAPI_BSPAttach***, определяя ПБУ, указанного в параметре *BSPUuid*.

ПБУ должен проверить совместимость с версией, указанной в параметре *Version*. Если версия несовместима, то функция возвращает отказ. ПБУ должен выполнить все инициализации, требуемые для поддержки нового вызова.

ПБУ должен подключить специализированные модули БиоАПИ, если они им поддерживаются.

Примечание — Данная функция аналогична функции ***BioAPI_BSPAttach*** (см. 8.1.7).

9.3.1.3.2 Параметры

BSPUuid (входной) — указатель на УУИД вызываемого ПБУ.

Version (входной) — номера редакции и поправки данной спецификации БиоАПИ, которую приложение предлагает ПБУ для поддержки. ПБУ должен определить, совместим ли он с требуемой версией.

UnitList (входной) — указатель на буфер, содержащий список структур BioAPI_UNIT_LIST_ELEMENT, указывающих, какие модули БиоАПИ (поддерживаемые ПБУ) должны быть использованы в данной присоединенной сессии. Для каждой категории модуля БиоАПИ должно выполняться одно из следующих действий:

a) выбор специализированного модуля БиоАПИ: специализированный модуль БиоАПИ, который должен быть использован в данной присоединенной сессии, определяется добавлением ИД и категории;

b) выбор любого модуля БиоАПИ: если параметр *UnitID* в конкретном элементе устанавливают в BioAPI_DONT_CARE, ПБУ осуществляет выбор, какой модуль данной категории может быть использован, или возвращает ошибку, если он не поддерживает ни один из модулей данной категории. Если конкретная категория не указана, ПБУ выбирает поддерживаемый модуль БиоАПИ этой категории, но не возвращает ошибку, если он не поддерживает этот модуль;

c) выбор без указания модуля БиоАПИ: если параметр *UnitID* установлен в BioAPI_DONT_INCLUDE, ПБУ не присоединит модуль БиоАПИ данной категории, даже если он поддерживает ее.

Примечание — Любые соответствующие вызовы, требующие использования модуля БиоАПИ данной категории, закончатся возвращением ошибки.

NumUnits (входной) — число элементов модулей БиоАПИ в списке, на который указывает указатель *UnitList*. Если данный параметр содержит нулевое значение, то ПБУ выбирает модуль БиоАПИ для всех категорий модулей БиоАПИ, которыми ПБУ управляет непосредственно или косвенно.

BSPHandle (входной) — значение BioAPI_HANDLE, определенное инфраструктурой и связанное с присоединенной сессией, создаваемой данной функцией.

Примечание — Только один модуль БиоАПИ каждой категории может быть одновременно соотнесен с прикрепленной сессией.

9.3.1.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

9.3.1.3.4 Ошибки

Аналогичны указанным для функции ***BioAPI_BSPAttach*** (8.1.7.4).

9.3.1.4 BioSPI_BSPDetach

```
BioAPI_RETURN BioAPI BioSPI_BSPDetach
  (BioAPI_HANDLE BSPHandle);
```

9.3.1.4.1 Описание

Данная функция вызывается инфраструктурой БиоАПИ один раз при каждом вызове функции ***BioAPI_BSPDetach,*** определяющей присоединенную сессию, указанную в *BSPHandle*. ПБУ должен выполнить все операции по завершению работы, связанные со специализированным присоединенным дескриптором.

Примечание — Данная функция аналогична функции ***BioAPI_BSPDetach*** (см. 8.1.8).

9.3.1.4.2 Параметры

BSPHandle (входной) — дескриптор, связанный с присоединенной сессией, завершаемой данной функцией.

9.3.1.4.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

9.3.1.4.4 Ошибки

Аналогичны указанным для функции ***BioAPI_BSPDetach*** (8.1.8.4).

9.3.1.5 BioSPI_QueryUnits

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

Примечание — Подробное описание функции приведено в 8.1.9 (BioAPI_QueryUnits).

9.3.1.6 BioSPI_QueryBFPs

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

Примечание 1 — При вызове данной функции ПБУ может использовать обратный вызов обработчика перечислений ПБФ (9.2.2) для получения информации обо всех установленных ПБФ и затем может создать список всех поддерживаемых ПБФ путем проверки каждой записи массива, возвращаемого данным обратным вызовом.

Примечание 2 — Подробное описание функции приведено в 8.1.11 (BioAPI_QueryBFPs).

9.3.1.7 BioSPI_ControlUnit

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

Примечание — Подробное описание функции приведено в 8.1.12 (BioAPI_ControlUnit).

9.3.2 Операции над дескриптором данных ИПУ

9.3.2.1 BioSPI_FreeBIRHandle

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

Примечание — Подробное описание функции приведено в 8.2.1 (BioAPI_FreeBIRHandle).

9.3.2.2 BioSPI_GetBIRFromHandle

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

Примечание — Подробное описание функции приведено в 8.2.2 (BioAPI_GetBIRFromHandle).

9.3.2.3 BioSPI_GetHeaderFromHandle

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

Примечание — Подробное описание функции приведено в 8.2.3 (BioAPI_GETHeaderFromHandle).

9.3.3 Обратные вызовы и работа с событиями ИПУ

9.3.3.1 BioSPI_EnableEvents

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

Примечание — Подробное описание функции приведено в 8.3.1 (BioAPI_EnableEvents).

9.3.3.2 BioSPI_SetGUICallbacks

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

Примечание — Подробное описание функции приведено в 8.3.2 (BioAPI_SetGUICallbacks).

9.3.4 Биометрические операции ИПУ

9.3.4.1 BioSPI_Capture

```
BioAPI_RETURN BioAPI BioSPI_Capture
  (BioAPI_HANDLE BSPHandle,
  BioAPI_BIR_PURPOSE Purpose,
  BioAPI_BIR_SUBTYPE Subtype,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
  BioAPI_BIR_HANDLE *CapturedBIR,
  int32_t Timeout,
  BioAPI_BIR_HANDLE *AuditData);
```

Примечание — Подробное описание функции приведено в 8.4.1 (BioAPI_Capture).

9.3.4.2 BioSPI_CreateTemplate

```
BioAPI_RETURN BioAPI BioSPI_CreateTemplate
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_INPUT_BIR *CapturedBIR,
  const BioAPI_INPUT_BIR *ReferenceTemplate,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
  BioAPI_BIR_HANDLE *NewTemplate,
  const BioAPI_DATA *Payload,
  BioAPI_UUID *TemplateUUID);
```

Примечание — Подробное описание функции приведено в 8.4.2 (BioAPI_CreateTemplate).

9.3.4.3 BioSPI_Process

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

Примечание — Подробное описание функции приведено в 8.4.3 (BioAPI_Process).

9.3.4.4 BioSPI_ProcessWithAuxBIR

```
BioSPI_RETURN BioAPI BioSPI_ProcessWithAuxBIR
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_INPUT_BIR *CapturedBIR,
  const BioAPI_INPUT_BIR *AuxiliaryData,
```

```
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
  BioAPI_BIR_HANDLE *ProcessedBIR);
```

Примечание — Подробное описание функции приведено в 8.4.4 (BioAPI_ProcessWithAuxBIR).

9.3.4.5 BioSPI_VerifyMatch

```
BioAPI_RETURN BioAPI BioSPI_VerifyMatch
  (BioAPI_HANDLE BSPHandle,
  BioAPI_FMR MaxFMRRequested,
  const BioAPI_INPUT_BIR *ProcessedBIR,
  const BioAPI_INPUT_BIR *ReferenceTemplate,
  BioAPI_BIR_HANDLE *AdaptedBIR,
  BioAPI_BOOL *Result,
  BioAPI_FMR *FMRAchieved,
  BioAPI_DATA *Payload);
```

Примечание — Подробное описание функции приведено в 8.4.5 (BioAPI_VerifyMatch).

9.3.4.6 BioSPI_IdentifyMatch

```
BioAPI_RETURN BioAPI BioSPI_IdentifyMatch
  (BioAPI_HANDLE BSPHandle,
  BioAPI_FMR MaxFMRRequested,
  const BioAPI_INPUT_BIR *ProcessedBIR,
  const BioAPI_IDENTIFY_POPULATION *Population,
  uint32_t TotalNumberOfTemplates,
  BioAPI_BOOL Binning,
  uint32_t MaxNumberOfResults,
  uint32_t *NumberOfResults,
  BioAPI_CANDIDATE **Candidates,
  int32_t Timeout);
```

Примечание — Подробное описание функции приведено в 8.4.6 (BioAPI_IdentifyMatch).

9.3.4.7 BioSPI_Enroll

```
BioAPI_RETURN BioAPI BioSPI_Enroll
  (BioAPI_HANDLE BSPHandle,
  BioAPI_BIR_PURPOSE Purpose,
  BioAPI_BIR_SUBTYPE Subtype,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
  const BioAPI_INPUT_BIR *ReferenceTemplate,
  BioAPI_BIR_HANDLE *NewTemplate,
  const BioAPI_DATA *Payload,
  int32_t Timeout,
  BioAPI_BIR_HANDLE *AuditData,
  BioAPI_UUID *TemplateUUID);
```

Примечание — Подробное описание функции приведено в 8.4.7 (BioAPI_Enroll).

9.3.4.8 BioSPI_Verify

```
BioAPI_RETURN BioAPI BioSPI_Verify
  (BioAPI_HANDLE BSPHandle,
  BioAPI_FMR MaxFMRRequested,
  const BioAPI_INPUT_BIR *ReferenceTemplate,
  BioAPI_BIR_SUBTYPE Subtype,
  BioAPI_BIR_HANDLE *AdaptedBIR,
  BioAPI_BOOL *Result,
  BioAPI_FMR *FMRAchieved,
  BioAPI_DATA *Payload,
  int32_t Timeout,
  BioAPI_BIR_HANDLE *AuditData);
```

Примечание — Подробное описание функции приведено в 8.4.8 (BioAPI_Verify).

9.3.4.9 BioSPI_Identify

```
BioAPI_RETURN BioAPI BioSPI_Identify
  (BioAPI_HANDLE BSPHandle,
  BioAPI_FMR MaxFMRRequested,
  BioAPI_BIR_SUBTYPE Subtype,
  const BioAPI_IDENTIFY_POPULATION *Population,
  uint32_t TotalNumberOfTemplates,
  BioAPI_BOOL Binning,
  uint32_t MaxNumberOfResults,
  uint32_t *NumberOfResults,
  BioAPI_CANDIDATE **Candidates,
  int32_t Timeout,
  BioAPI_BIR_HANDLE *AuditData);
```

Примечание — Подробное описание функции приведено в 8.4.9 (BioAPI_Identify).

9.3.4.10 BioSPI_Import

```
BioAPI_RETURN BioAPI BioSPI_Import
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_DATA *InputData,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *InputFormat,
  const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
  BioAPI_BIR_PURPOSE Purpose,
  BioAPI_BIR_HANDLE *ConstructedBIR);
```

Примечание — Подробное описание функции приведено в 8.4.10 (BioAPI_Import).

9.3.4.11 BioSPI_PresetIdentifyPopulation

```
BioAPI_RETURN BioAPI BioAPI_PresetIdentifyPopulation
(BioAPI_HANDLE BSPHandle,
const BioAPI_IDENTIFY_POPULATION *Population);
```

Примечание — Подробное описание функции приведено в 8.4.11 (BioAPI_PresetIdentifyPopulation).

9.3.5 Операции над базой данных ИПУ

9.3.5.1 BioSPI_DbOpen

```
BioAPI_RETURN BioAPI BioSPI_DbOpen
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_UUID *DbUuid,
  BioAPI_DB_ACCESS_TYPE AccessRequest,
  BioAPI_DB_HANDLE *DbHandle,
  BioAPI_DB_MARKER_HANDLE *MarkerHandle);
```

Примечание — Подробное описание функции приведено в 8.5.1 (BioAPI_DbOpen).

9.3.5.2 BioSPI_DbClose

```
BioAPI_RETURN BioAPI BioSPI_DbClose
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle);
```

Примечание — Подробное описание функции приведено в 8.5.2 (BioAPI_DbClose).

9.3.5.3 BioSPI_DbCreate

```
BioAPI_RETURN BioAPI BioSPI_DbCreate
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_UUID *DbUuid,
  uint32_t NumberOfRecords,
  BioAPI_DB_ACCESS_TYPE AccessRequest,
  BioAPI_DB_HANDLE *DbHandle);
```

Примечание — Подробное описание функции приведено в 8.5.3 (BioAPI_DbCreate).

9.3.5.4 BioSPI_DbDelete

```
BioAPI_RETURN BioAPI BioSPI_DbDelete
(BioAPI_HANDLE BSPHandle,
const BioAPI_UUID *DbUuid);
```

Примечание — Подробное описание функции приведено в 8.5.4 (BioAPI_DbDelete).

9.3.5.5 BioSPI_DbSetMarker

```
BioAPI_RETURN BioAPI BioSPI_DbSetMarker
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle,
  const BioAPI_UUID *KeyValue,
  BioAPI_DB_MARKER_HANDLE MarkerHandle);
```

Примечание — Подробное описание функции приведено в 8.5.5 (BioAPI_DbSetMarker).

9.3.5.6 BioSPI_DbFreeMarker

```
BioAPI_RETURN BioAPI BioSPI_DbFreeMarker
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_MARKER_HANDLE MarkerHandle);
```

Примечание — Подробное описание функции приведено в 8.5.6 (BioAPI_DbFreeMarker).

9.3.5.7 BioSPI_DbStoreBIR

```
BioAPI_RETURN BioAPI BioSPI_DbStoreBIR
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_INPUT_BIR *BIRToStore,
  BioAPI_DB_HANDLE DbHandle,
  BioAPI_UUID_PTR Uuid);
```

Примечание — Подробное описание функции приведено в 8.5.7 (BioAPI_DbStoreBIR).

9.3.5.8 BioSPI_DbGetBIR

```
BioAPI_RETURN BioAPI BioSPI_DbGetBIR
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle,
  const BioAPI_UUID *KeyValue,
  BioAPI_BIR_HANDLE *RetrievedBIR,
  BioAPI_DB_MARKER_HANDLE *MarkerHandle);
```

Примечание — Подробное описание функции приведено в 8.5.8 (BioAPI_DbGetBIR).

9.3.5.9 BioSPI_DbGetNextBIR

```
BioAPI_RETURN BioAPI BioSPI_DbGetNextBIR
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle,
  BioAPI_DB_MARKER_HANDLE MarkerHandle,
  BioAPI_BIR_HANDLE *RetrievedBIR,
  BioAPI_UUID *BirUuid);
```

Примечание — Подробное описание функции приведено в 8.5.9 (BioAPI_DbGetNextBIR).

9.3.5.10 BioSPI_DbDeleteBIR

```
BioAPI_RETURN BioAPI BioSPI_DbDeleteBIR
  (BioAPI_HANDLE BSPHandle,
  BioAPI_DB_HANDLE DbHandle,
  const BioAPI_UUID *KeyValue);
```

Примечание — Подробное описание функции приведено в 8.5.10 (BioAPI_DbDeleteBIR).

9.3.6 Операции с модулями БиоАПИ ИПУ

9.3.6.1 BioSPI_SetPowerMode

```
BioAPI_RETURN BioAPI BioSPI_SetPowerMode
  (BioAPI_HANDLE BSPHandle,
  BioAPI_UNIT_ID UnitId,
  BioAPI_POWER_MODE PowerMode);
```

Примечание — Подробное описание функции приведено в 8.6.1 (BioAPI_SetPowerMode).

9.3.6.2 BioSPI_SetIndicatorStatus

```
BioAPI_RETURN BioAPI BioSPI_SetIndicatorStatus
  (BioAPI_HANDLE BSPHandle,
  BioAPI_UNIT_ID UnitId,
  BioAPI_INDICATOR_STATUS IndicatorStatus);
```

Примечание — Подробное описание функции приведено в 8.6.2 (BioAPI_SetIndicatorStatus).

9.3.6.3 BioSPI_GetIndicatorStatus

```
BioAPI_RETURN BioAPI BioSPI_GetIndicatorStatus
  (BioAPI_HANDLE BSPHandle,
  BioAPI_UNIT_ID UnitId,
  BioAPI_INDICATOR_STATUS *IndicatorStatus);
```

Примечание — Подробное описание функции приведено в 8.6.3 (BioAPI_GetIndicatorStatus).

9.3.6.4 BioSPI_CalibrateSensor

```
BioAPI_RETURN BioAPI BioSPI_CalibrateSensor
  (BioAPI_HANDLE BSPHandle,
  int32_t Timeout);
```

Примечание — Подробное описание функции приведено в 8.6.4 (BioAPI_CalibrateSensor).

9.3.7 Служебные операции ИПУ

9.3.7.1 BioSPI_Cancel

```
BioAPI_RETURN BioAPI BioSPI_Cancel
  (BioAPI_HANDLE BSPHandle);
```

Примечание — Подробное описание функции приведено в 8.7.1 (BioAPI_Cancel).

9.3.7.2 BioSPI_Free

```
BioAPI_RETURN BioAPI BioSPI_Free
  (void* Ptr);
```

Примечание — Подробное описание функции приведено в 8.7.2 (BioAPI_Free).

10 Интерфейс реестра компонентов

Реестр компонентов хранит статическую информацию, описывающую возможности компонентов БиоАПИ. Для получения данной информации биометрические приложения опрашивают реестр компонентов. Реестр компонентов является частью инфраструктуры БиоАПИ.

Приложение, выполняющее установку компонентов БиоАПИ (инфраструктуру БиоАПИ, ПБУ или ПБФ), размещает информацию о них в реестр компонентов. Данная информация используется биометрическим приложением для определения того, какие ПБУ и ПБФ были установлены. Биометрическое приложение может использовать данную информацию динамически в своей решающей логике. Например, оно может решить, необходимо ли производить какой-либо (необязательный) вызов функции к данному ПБУ, в зависимости от записи в реестре для данного ПБУ о поддержке вызова. Также реестр компонентов предоставляет информацию относительно форматов ББД, поддерживаемых ПБУ (см. 6.7 и приложение В), и значения по умолчанию ПБУ для различных общих параметров (таких, как время ожидания).

Примечание — Информация о модулях БиоАПИ не хранится в реестре компонентов, но предоставляется при загрузке ПБУ (как событие вставки) или в ответ на запрос (BioAPI_QueryUnits).

Реестр компонентов разработан как независимый от платформы, но может использовать встроенный реестр какой-либо операционной системы (такой, как системный реестр Microsoft Windows™).

Средства регистрации информации в реестре компонентов приведены в 10.2.

10.1 Схема реестра компонентов БиоАПИ

Все компоненты биометрической системы, выполненной в соответствии с БиоАПИ, должны быть записаны в реестре компонентов. Данные записи не зависят от операционной системы. Данный подраздел определяет информацию, хранимую в реестре для каждого компонента биометрической системы, основанной на БиоАПИ.

Примечание — Все строковые типы, определенные как элементы структуры компонентов, должны содержать символы, закодированные по ИСО/МЭК 10646 (формат UTF-8), если не установлены иные требования для структур данных, указанных в разделе 7.

10.1.1 Схема инфраструктуры

Данная схема содержит атрибуты инфраструктуры БиоАПИ.

Таблица 2 — Элементы схемы инфраструктуры

Имя поля	Тип данных поля	Описание
FrameworkUUID	UINT8_T [16]	УУИД, однозначно идентифицирующий инфраструктуру БиоАПИ
Description	STRING	Описание продукции инфраструктуры БиоАПИ, представленное в виде текста
Path	STRING	Путь (включая имя файла) к исполняемому коду инфраструктуры БиоАПИ
Spec Version	UINT8_T[2]	Поддерживаемая версия спецификации инфраструктуры БиоАПИ
Product Version	STRING	Версия программного продукта инфраструктуры БиоАПИ
Vendor	STRING	Наименование изготовителя инфраструктуры, представленное в виде текста
Description	STRING	Описание программного продукта инфраструктуры БиоАПИ, представленное в виде текста
FW Property ID	UINT8_T [16]	УУИД, определяющий формат элемента свойств инфраструктуры (назначенный изготовителем)
FW Property	DATA	Специализированная информация изготовителя об инфраструктуре

Примечание — См. также 7.30.

10.1.2 Схема ПБУ

Данная схема описывает возможности ПБУ. Для каждого установленного в биометрическую систему ПБУ существует одна схема.

Таблица 3 — Элементы схемы ПБУ

Имя поля	Тип данных поля	Описание
BSPUUID	UINT8_T[16]	УУИД, однозначно идентифицирующий ПБУ
Description	STRING	Текстовое описательное имя ПБУ
Path	STRING	Путь к исполняемому ПБУ, включающий имя файла исполняемого кода компонента ПБУ (может быть URL-адресом)
Spec Version	UINT8_T	Версия поддерживаемой спецификации БиоАПИ
Product Version	STRING	Версия программного продукта ПБУ
Vendor	STRING	Наименование изготовителя ПБУ, представленное в виде текста
BSP Supported Formats	MULTIUINT32_T	Массив 2-байтовых пар в виде целых чисел. Каждая пара определяет поддерживаемый формат ББД (см. BioAPI_BIR_BIOMETRIC_DATA_FORMAT)

Продолжение таблицы 3

Имя поля	Тип данных поля	Описание
Number of Supported Formats	UINT32_T	Целое число, определяющее число форматов ББД, поддерживаемых ПБУ (число 2-байтовых пар в виде целых чисел в поле BSP Supported Formats)
Factors Mask	UINT32_T	Маска, указывающая, какие формы определения подлинности поддерживаются ПБУ (см. BioAPI_BIR_BIOMETRIC_TYPE)
Operations	UINT32_T	Операции (функции), поддерживаемые ПБУ (см. BioAPI_OPERATIONS_MASK)
Options	UINT32_T	Опции, поддерживаемые ПБУ (см. BioAPI_OPTIONS_MASK)
Payload Policy	UINT32_T	Значение порога (минимальное значение ОЛС), используемое для определения возможности предоставления полезной информации (см. BioAPI_FMR)
Max Payload Size	UINT32_T	Максимальный размер полезной информации в байтах
Default Verify Timeout	INT32_T	Значение времени ожидания по умолчанию в миллисекундах, используемое ПБУ для операций верификации в случае, если время ожидания не установлено биометрическим приложением
Default Identify Timeout	INT32_T	Значение времени ожидания по умолчанию в миллисекундах, используемое ПБУ для операций идентификации в случае, если время ожидания не установлено биометрическим приложением
Default Capture Timeout	INT32_T	Значение времени ожидания по умолчанию в миллисекундах, используемое ПБУ для операций получения данных в случае, если время ожидания не установлено биометрическим приложением
Default Enroll Timeout	INT32_T	Значение времени ожидания по умолчанию в миллисекундах, используемое ПБУ для операций регистрации в случае, если время ожидания не установлено биометрическим приложением
Default calibrate Timeout	INT32_T	Значение времени ожидания по умолчанию в миллисекундах, используемое ПБУ для операций калибровки в случае, если время ожидания не установлено биометрическим приложением
MAX BSP DB size	UINT32_T	Максимальный размер базы данных ЗБИ. Нулевое значение указывает на отсутствие базы данных для этого ПБУ[1)]
Max Identify Population	UINT32_T	Наибольшая совокупность, поддерживаемая функцией идентификации [1)]. Значение 0xFFFFFFFF указывает на отсутствие ограничений

[1)] Применяется только в том случае, если ПБУ может непосредственно управлять отдельным модулем архива. Значение 0 означает, что ПБУ способен управлять несколькими модулями (либо напрямую, либо с помощью интерфейса ПБФ) и информация об этих модулях будет предоставлена как часть уведомления о вставке (часть схемы модуля).

Примечание — См. также 7.16.

10.1.3 Схема ПБФ

Данная схема описывает возможности ПБФ. Для каждого установленного в биометрической системе ПБФ существует одна схема.

Таблица 4 — Элементы схемы ПБФ

Имя поля	Тип данных поля	Описание
BFP UUID	UINT8_T [16]	УУИД, однозначно идентифицирующий ПБФ
BFP Category	UINT32_T	Категория ПБФ, определенная УУИД ПБФ
Description	STRING	Текстовое описательное имя ПБФ
Path	STRING	Путь к исполняемому ПБФ, включающий в себя имя файла исполняемого кода ПБФ (может быть URL-адресом)
Spec Version	UINT8_T	Версия поддерживаемого стандарта BioAPI[1)]
Product Version	STRING	Версия программного продукта ПБФ
Vendor	STRING	Наименование изготовителя ПБФ, представленное в виде текста
BFP Supported Formats	MULTIUINT32_T	Массив 2-байтовых пар в виде целых чисел. Каждая пара определяет поддерживаемый формат ББД (см. BioAPI_BIR_BIOMETRIC_DATA_FORMAT)
Number of Supported Formats	UINT32_T	Целое число, определяющее число форматов ББД, поддерживаемых ПБФ (число 2-байтовых пар в виде целых чисел в поле BFP Supported Formats)
Factors Mask	UINT32_T	Маска, указывающая, какие формы установления подлинности поддерживаются ПБФ (см. BioAPI_BIR_BIOMETRIC_TYPE)
BFP Property ID	UINT8_T [16]	УУИД, идентифицирующий формат поля BFP Properties (указывает изготовитель)
BFP Property	DATA	Специализированная информация изготовителя о ПБФ

[1)] Спецификация интерфейса поставщика функции будет установлена в отдельной части комплекса стандартов ИСО/МЭК 19784.

Примечание — См. также 7.3.

10.2 Функции реестра компонентов

Для упрощения разработки совместимых с БиоАПИ приложений и ПБУ используют служебные функции реестра компонентов, приведенные в настоящем разделе.

Функции установки предусмотрены для того, чтобы конструктор ПБУ и ПБФ мог заполнить схемы ПБУ и ПБФ реестра компонентов. Реестр компонентов не зависит от операционной системы.

10.2.1 BioAPI_Util_InstallBsp

10.2.1.1 Описание

Данная функция устанавливает, модифицирует (обновляет) или удаляет ссылку на ПБУ в реестре компонентов.

```
BioAPI_RETURN BioAPI_Util_InstallBsp
  (BioAPI_INSTALL_ACTION Action,
  BioAPI_INSTALL_ERROR *pError,
  BioAPI_BSP_SCHEMA_PTR pBSPSchema);
```

После инициализации данной функции для установки инфраструктура БиоАПИ должна создать запись схемы ПБУ в реестре компонентов и включить в него содержание входной схемы ПБУ.

После инициализации данной функции для обновления инфраструктура БиоАПИ должна заменить существующие записи схемы ПБУ в реестре компонентов содержанием входной схемы ПБУ, основываясь на УУИД ПБУ в данной схеме.

После инициализации данной функции для удаления инфраструктура БиоАПИ должна удалить запись схемы ПБУ из реестра компонентов для ПБУ, обозначенного параметром *BSPUuid* во входной схеме.

10.2.1.2 Параметры

Action (входной) — определяет тип выполняемого действия. Допустимыми значениями являются следующие:

Значение	Описание
BioAPI_INSTALL_ACTION_INSTALL	— установка ПБУ (добавление в реестр);
BioAPI_INSTALL_ACTION_REFRESH	— обновление информации в реестре компонентов;
BioAPI_INSTALL_ACTION_UNINSTALL	— удаление ПБУ из реестра.

Error (выходной) — указатель на структуру BioAPI_INSTALL_ERROR. Если выполнение функции заканчивается ошибкой, данная структура содержит дополнительную информацию о типе ошибки.

BSPSchema (входной/выходной) — указатель на элементы схемы ПБУ, определенные в 7.16, описывающие свойства и характеристики устанавливаемого ПБУ. При вызове функции с параметром *Action*, установленным в BioAPI_INSTALL_ACTION_UNINSTALL, необходимо задать только элемент *BSPUuid* параметра *BSPSchema*.

10.2.1.3 Возвращаемые значения

Если функция выполнена успешно, то она возвращает BioAPI_OK. Если функция завершается неудачей, она возвращает код ошибки БиоАПИ (см. раздел 11). Параметр *Error* содержит дополнительные сведения об ошибке.

10.2.2 BioAPI_Util_InstallBFP

10.2.2.1 Описание

Данная функция устанавливает, модифицирует (обновляет) или удаляет ссылку на ПБФ в реестре компонентов.

```
BioAPI_RETURN BioAPI_Util_InstallUnit
  (BioAPI_INSTALL_ACTION Action,
  BioAPI_INSTALL_ERROR *Error,
  const BioAPI_BFP_SCHEMA *BFPSchema);
```

10.2.2.2 Параметры

Action (входной) — определяет тип выполняемого действия. Допустимыми значениями являются следующие:

Значение	Описание
BioAPI_INSTALL_ACTION_INSTALL	— установка ПБФ (добавление записи в реестр);
BioAPI_INSTALL_ACTION_REFRESH	— обновление информации в реестре компонентов;
BioAPI_INSTALL_ACTION_UNINSTALL	— удаление ПБФ (удаление записи из реестра).

Error (выходной) — указатель на структуру BioAPI_INSTALL_ERROR. Если выполнение функции заканчивается ошибкой, то данная структура содержит дополнительную информацию об ошибке.

BFPSchema (входной/выходной) — указатель на элемент реестра компонентов, описывающий свойства и характеристики устанавливаемого ПБФ (см. 7.3). При вызове функции с параметром *Action*, установленным в BioAPI_INSTALL_ACTION_UNINSTALL, необходимо задать только элемент *BFPUuid* параметра *BFPSchema*.

10.2.2.3 Возвращаемые значения

Если функция выполнена успешно, она возвращает BioAPI_OK. Если функция завершается неудачей, она возвращает код ошибки БиоАПИ. Параметр *Error* содержит дополнительные сведения об ошибке.

11 Обработка ошибок БиоАПИ

Все функции БиоАПИ возвращают значение типа BioAPI_RETURN. Значение BioAPI_OK указывает на успешное выполнение функции. Любое другое значение указывает на наличие ошибки. В настоящем стандарте установлены ограничения на возвращаемые значения ошибок, возникающих в инфраструктуре БиоАПИ, в ПБУ или в модулях БиоАПИ, присоединенных к ПБУ.

11.1 Значения ошибок и схемы кодов ошибок

Значение ошибки представляет собой полное 32-разрядное значение BioAPI_RETURN.

Кодом ошибки являются младшие 24 бита значения ошибки и идентифицируют фактические условия ошибки.

Восемь старших битов значения ошибки идентифицируют источник ошибки:

a) инфраструктура БиоАПИ;

b) ПБУ;

c) модуль БиоАПИ, присоединенный к ПБУ.

11.2 Перечень кодов ошибок и значений ошибок

11.2.1 Константы значения ошибки БиоАПИ

#define BIOAPI_FRAMEWORK_ERROR (0x00000000)
#define BIOAPI_BSP_ERROR (0x01000000)
#define BIOAPI_UNIT_ERROR (0x02000000)

11.2.2 Определяемые реализацией коды ошибок

Коды ошибок 0x000000 — 0x0000FF зарезервированы.

11.2.3 Общие коды ошибки

#define BioAPIERR_INTERNAL_ERROR (0x000101)

Общая системная ошибка указывает, что произошла ошибка операционной системы или ошибка внутреннего состояния и состояние системы может быть неизвестно.

#define BioAPIERR_MEMORY_ERROR (0x000102)

Ошибка памяти.

#define BioAPIERR_INVALID_POINTER (0x000103)

Входной / выходной параметр функции или входное / выходное поле структуры данных является недействительным указателем.

#define BioAPIERR_INVALID_INPUT_POINTER (0x000104)

Входной параметр функции или входное поле структуры данных является недействительным указателем.

#define BioAPIERR_INVALID_OUTPUT_POINTER (0x000105)

Выходной параметр функции или выходное поле структуры данных является недействительным указателем.

#define BioAPIERR_FUNCTION_NOT_SUPPORTED (0x000106)

Функция не реализована поставщиком биометрической услуги.

#define BioAPIERR_OS_ACCESS_DENIED (0x000107)

Операционная система запрещает доступ к требуемому ресурсу.

#define BioAPIERR_FUNCTION_FAILED (0x000108)

Функция не выполнена по неизвестной причине.

#define BioAPIERR_INVALID_UUID (0x000109)

Входной УУИД недействителен. Данный тип ошибки может появляться в случае, если компонент, запрашиваемый данным УУИД, не присутствует в системе или не может быть найден.

#define BioAPIERR_INCOMPATIBLE_VERSION (0x00010a)

Ошибка вследствие несовместимости версии. Возникает, если вызываемый компонент не поддерживает ожидаемую приложением версию БиоАПИ.

#define BioAPIERR_INVALID_DATA (0x00010b)

Данные во входном параметре недействительны.

#define BioAPIERR_UNABLE_TO_CAPTURE (0x00010c)

Связанный ПБУ не может получить исходные образцы от запрашиваемого модуля БиоАПИ.

#define BioAPIERR_TOO_MANY_HANDLES (0x00010d)

Связанный ПБУ не имеет более ресурсов для размещения дескрипторов ЗБИ.

#define BioAPIERR_TIMEOUT_EXPIRED (0x00010e)

Функция завершена по причине истечения времени ожидания.

#define BioAPIERR_INVALID_BIR (0x00010f)

Входная ЗБИ непригодна для требуемого назначения.

#define BioAPIERR_BIR_SIGNATURE_FAILURE (0x000110)

Связанный ПБУ не смог подтвердить достоверность подписи ЗБИ.

#define BioAPIERR_UNABLE_TO_STORE_PAYLOAD (0x000111)

Связанный ПБУ не может сохранить полезную информацию.

#define BioAPIERR_NO_INPUT_BIRS (0x000112)

Совокупность для идентификации является пустой.

#define BioAPIERR_UNSUPPORTED_FORMAT (0x000113)

Связанный ПБУ не поддерживает запрашиваемый формат ББД.

#define BioAPIERR_UNABLE_TO_IMPORT (0x000114)

Связанный ПБУ не может создать ЗБИ из входных данных.

#define BioAPIERR_INCONSISTENT_PURPOSE (0x000115)

Назначение, записанное в ЗБИ, и требуемое назначение несовместимы с выполняемой функцией.

#define BioAPIERR_BIR_NOT_FULLY_PROCESSED (0x000116)

Функция требует полностью обработанную ЗБИ.

#define BioAPIERR_PURPOSE_NOT_SUPPORTED (0x000117)

ПБУ не поддерживает требуемое назначение.

#define BioAPIERR_USER_CANCELLED (0x000118)

Пользователь отменил операцию до ее завершения или истечения времени ожидания.

#define BioAPIERR_UNIT_IN_USE (0x000119)

ПБУ или модуль БиоАПИ, присоединенный к ПБУ, в данный момент используется другим биометрическим приложением.

#define BioAPIERR_INVALID_BSP_HANDLE (0x00011a)

Данный дескриптор ПБУ является недействительным.

#define BioAPIERR_FRAMEWORK_NOT_INITIALIZED (0x00011b)

Вызов функции без инициализации инфраструктуры БиоАПИ.

#define BioAPIERR_INVALID_BIR_HANDLE (0x00011c)

Дескриптор ЗБИ недействителен, то есть не существует или был удален.

#define BioAPIERR_CALIBRATION_NOT_SUCCESSFUL (0x00011d)

Попытка калибровки модуля датчиков не может быть завершена удачно.

#define BioAPIERR_PRESET_BIR_DOES_NOT_EXIST (0x00011e)

Совокупность ЗБИ не была предустановлена.

#define BioAPIERR_BIR_DECRYPTION_FAILURE (0x00011f)

ПБУ не может расшифровать входную ЗБИ и, следовательно, не может использовать ее для запрашиваемой операции.

11.2.4 Коды ошибок управления компонентом

#define BioAPIERR_COMPONENT_FILE_REF_NOT_FOUND (0x000201)

Ссылка на файл компонента не может быть получена.

#define BioAPIERR_BSP_LOAD_FAIL (0x000202)

Инфраструктура не способна успешно загрузить ПБУ.

#define BioAPIERR_BSP_NOT_LOADED (0x000203)

ПБУ, для которого запрашивается действие, не загружен.

#define BioAPIERR_UNIT_NOT_INSERTED (0x000204)

Модуль БиоАПИ, для которого запрашивается действие, не подключен.

#define BioAPIERR_INVALID_UNIT_ID (0x000205)

Запрашивается недействительный ИД модуля БиоАПИ.

#define BioAPIERR_INVALID_CATEGORY (0x000206)

Запрашивается недействительная категория ПБФ или модуля БиоАПИ.

11.2.5 Значения ошибок базы данных

#define BioAPIERR_INVALID_DB_HANDLE (0x000300)

Недействительный дескриптор базы данных.

#define BioAPIERR_UNABLE_TO_OPEN_DATABASE (0x000301)

Связанный ПБУ не может открыть указанную базу данных.

#define BioAPIERR_DATABASE_IS_LOCKED (0x000302)

База данных не может быть открыта для доступа из-за того, что является заблокированной.

#define BioAPIERR_DATABASE_DOES_NOT_EXIST (0x000303)

Указанной базы данных не существует.

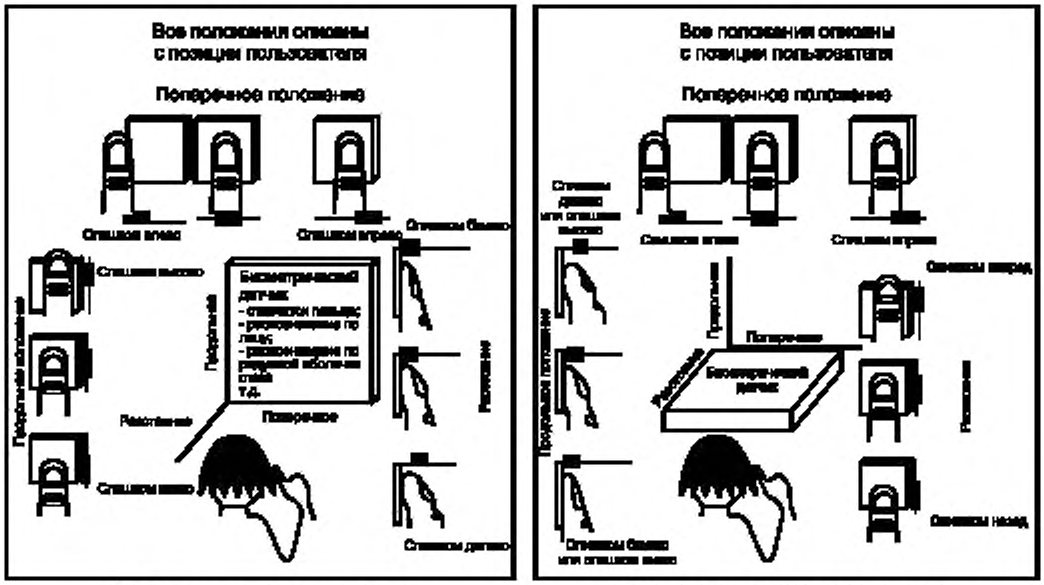
#define BioAPIERR_DATABASE_ALREADY_EXISTS (0x000304)

Отказ создания базы данных, так как она уже существует.

```
#define BioAPIERR_INVALID_DATABASE_NAME                    (0x000305)
Несуществующее имя базы данных (недействительный УУИД).
#define BioAPIERR_RECORD_NOT_FOUND                         (0x000306)
Не существует записи с запрашиваемым ключом.
#define BioAPIERR_MARKER_IS_INVALID                        (0x000307)
Указанный маркер недействителен.
#define BioAPIERR_DATABASE_IS_OPEN                         (0x000308)
База данных уже открыта.
#define BioAPIERR_INVALID_ACCESS_REQUEST                   (0x000309)
Неизвестный тип доступа.
#define BioAPIERR_END_OF_DATABASE                          (0x00030a)
Достигнут конец базы данных.
#define BioAPIERR_UNABLE_TO_CREATE_DATABASE                (0x00030b)
Связанный ПБУ не может создать базу данных.
#define BioAPIERR_UNABLE_TO_CLOSE_DATABASE                 (0x00030c)
Связанный ПБУ не может закрыть базу данных.
#define BioAPIERR_UNABLE_TO_DELETE_DATABASE                (0x00030d)
Связанный ПБУ не может удалить базу данных.
#define BioAPIERR_DATABASE_IS_CORRUPT                      (0x00030e)
Указанная база данных повреждена.
```

11.2.6 Значения ошибок положения

В различных биометрических технологиях, основанных на отпечатке пальца, лице, радужной оболочке, сетчатке глаза или других характеристиках, устройства получения биометрических данных (модули датчиков) имеют три измерения с позиции пользователя. Коды ошибок, представленные в данном пункте, определяют ошибки, которые могут быть вызваны некорректным размещением биометрической характеристики пользователя в биометрическом устройстве при попытке получения биометрического образца. Примеры расположения датчика отпечатка пальца, который может быть установлен вертикально или горизонтально, приведены на рисунке 4. При неправильном размещении биометрической характеристики коды ошибок должны предоставлять пользователю необходимую информацию. Для вертикально расположенно-



а) Датчик расположен вертикально

б) Датчик расположен горизонтально

Рисунок 4 — Положение биометрического образца (на примере датчика отпечатка пальца)

го датчика сообщения о перемещении «вперед» и «назад» соответствуют интенсивности контакта или расстоянию до датчика, а для горизонтально расположенного датчика эти слова соответствуют продольным направлениям.

Если ПБУ или модулю БиоАПИ, генерирующему код ошибки, известна ориентация устройства получения биометрической характеристики, то он может выбрать код ошибки, правильно описывающий положение пользователя. В другом случае необходимо использовать более общие коды ошибок.

Примечание — Подобные ошибки должны быть возвращены функциями ***BioAPI_CreateTemplate, BioAPI_Process, BioAPI_ProcessWithAuxBIR, BioAPI_VerifyMatch*** и ***BioAPI_IdentifyMatch.*** Управляемый ПБУ ГИП других биометрических функций (***BioAPI_Capture, BioAPI_Enroll, BioAPI_Verify,*** и ***BioAPI_Identify***) сообщает пользователю, какие действия должны быть выполнены для получения качественного образца.

Биометрические приложения могут использовать следующие коды ошибок, определения которых приведены ниже, для обратной связи с пользователем с целью получения образцов более высокого качества независимо от используемой биометрической технологии, если она имеет три измерения.

Если ПБУ или модуль БиоАПИ, генерирующий код ошибки, не имеют информации об ориентации датчика, то он должен возвратить общие коды ошибок.

11.2.6.1 Общие коды ошибок положения

#define BioAPIERR_LOCATION_ERROR (0x000400)

Общая ошибка положения.

#define BioAPIERR_OUT_OF_FRAME (0x000401)

Недействительное горизонтальное или вертикальное положение.

#define BioAPIERR_INVALID_CROSSWISE_POSITION (0x000402)

Недействительное поперечное положение.

#define BioAPIERR_INVALID_LENGTHWISE_POSITION (0x000403)

Недействительное продольное положение.

#define BioAPIERR_INVALID_DISTANCE (0x000404)

Недействительное расстояние.

11.2.6.2 Специальные коды ошибок положения

#define BioAPIERR_LOCATION_TOO_RIGHT (0x000405)

Положение сдвинуто слишком вправо.

#define BioAPIERR_LOCATION_TOO_LEFT (0x000406)

Положение сдвинуто слишком влево.

#define BioAPIERR_LOCATION_TOO_HIGH (0x000407)

Положение слишком высокое.

#define BioAPIERR_LOCATION_TOO_LOW (0x000408)

Положение слишком низкое.

#define BioAPIERR_LOCATION_TOO_FAR (0x000409)

Положение слишком далеко.

#define BioAPIERR_LOCATION_TOO_NEAR (0x00040a)

Положение слишком близко.

#define BioAPIERR_LOCATION_TOO_FORWARD (0x00040b)

Положение слишком близко (вперед).

#define BioAPIERR_LOCATION_TOO_BACKWARD (0x00040c)

Положение слишком далеко (назад).

11.2.7 Коды ошибок качества

#define BioAPIERR_QUALITY_ERROR (0x000501)

Качество образца слишком низкое для успешного выполнения операции.

Примечание — Ошибки качества могут быть возвращены любой функцией, получающей ЗБИ БиоАПИ в качестве входного параметра.

Приложение А
(обязательное)

Соответствие

А.1 Общие положения

А.1.1 Проверке на соответствие требованиям настоящего стандарта подвергают:

- биометрические приложения;
- инфраструктуру БиоАПИ;
- ПБУ, включающий в себя один из следующих подклассов:
  - БиоАПИ совместимый ПБУ для верификации;
  - БиоАПИ совместимый ПБУ для идентификации;
  - БиоАПИ совместимый ПБУ для получения данных;
  - БиоАПИ совместимый механизм верификации;
  - БиоАПИ совместимый механизм идентификации.

А.1.2 Требования соответствия для биометрических приложений, инфраструктуры БиоАПИ и ПБУ определены в А.2, А.3 и А.4 соответственно.

Примечание — В настоящем стандарте не рассматриваются вопросы соответствия ПБФ, однако они описаны в других частях комплекса стандартов ИСО/МЭК 19784. Интерфейсы модулей БиоАПИ не стандартизованы.

А.2 Совместимость биометрического приложения спецификации БиоАПИ

А.2.1 Для проверки соответствия настоящему стандарту биометрическое приложение должно произвести вызов каждой используемой функции БиоАПИ в соответствии с требованиями настоящего стандарта. Все входные параметры должны присутствовать и быть действительными. Приложение должно принимать все действительные выходные параметры и возвращаемые значения.

А.2.2 Минимальный набор функций, которые должны быть вызваны, не установлен. Несмотря на это биометрическое приложение должно соответствовать установленным зависимостям вызовов функций.

А.3 Совместимость инфраструктуры спецификации БиоАПИ

А.3.1 Связующим звеном между соответствующими БиоАПИ приложениями и ПБУ является компонент инфраструктуры БиоАПИ. Он предназначен для решения следующих общих задач:

a) загрузки/присоединения ПБУ;

b) управления ПБУ и ПБФ;

c) поддержки и управления реестром компонентов;

d) пересылки вызовов функций/трансляции ПИП — ИПУ;

e) обработки уведомлений о событиях ПБУ и отправки этих уведомлений о событиях обработчику событий в приложение, которое загрузило данный ПБУ;

g) поддержки вызовов ПИП, связанных с инсталляцией или деинсталляцией компонентов БиоАПИ, с соответствующим обновлением реестра компонентов;

f) поддержки запросов со стороны ПБУ об установленных ПБФ.

А.3.2 Для соответствия БиоАПИ вышеуказанный компонент инфраструктуры БиоАПИ должен:

a) предоставлять функции управления компонентом в соответствии с 8.1;

b) предоставлять данные для реестра компонентов в соответствии с разделом 10;

c) выполнять трансляцию функций ПИП, указанных в разделе 8, в соответствующие функции ИПУ, указанные в разделе 9;

d) соответствовать структурам данных, определенным в разделе 7, и кодам ошибок, определенным в разделе 11, при реализации перечислений a) — c);

e) обрабатывать уведомления о событиях и обратных вызовах в соответствии с 7.28, 8.3 и 9.2.

А.3.3 Соответствующая инфраструктура БиоАПИ должна поддерживать все опции, установленные настоящим стандартом, так как это обеспечивает возможность приложению и ПБУ реализовать любую из этих опций. Не определено никаких подгрупп функций соответствующей инфраструктуры.

А.4 Соответствие ПБУ спецификации БиоАПИ

Для соответствия настоящему стандарту ПБУ должны реализовывать функции, обязательные для ПБУ соответствующего подкласса, в соответствии с ИПУ, определенным в разделе 9. ПБУ должны соответствовать одному из соответствующих подклассов, указанных в А.1.1.

ПБУ должны принимать все действительные входные параметры и возвращать действительные выходные параметры. Необязательные возможности и возвращаемые параметры могут отсутствовать, но если они реализованы, то они должны соответствовать требованиям настоящего стандарта. Дополнительные параметры не требуются.

Инсталляционный процесс ПБУ должен представить совокупность всех необходимых записей реестра компонентов.

ПБУ должны иметь действительный и уникальный УУИД, связанный с определенным программным продуктом и версией ПБУ.

Примечание — УУИД может быть самогенерируемым (ИСО 9834-8) и может быть одним и тем же в сложной системе, где установлен тот же ПБУ продукт/версия.

ЗБИ, созданные ПБУ, должны соответствовать структурам данных, определенным в разделе 7. ПБУ должны возвращать только данные BioAPI_BIR, содержащие зарегистрированный FormatOwner со связанным действительным FormatID (7.6 и приложение В).

ПБУ должны выполнять обработку ошибок в соответствии с разделом 11.

Все ПБУ должны поддерживать основные операции управления компонентом (9.3.1), операции над дескрипторами (9.3.2), событиями (9.3.3.1) и утилитами (9.3.7). Обратные вызовы (9.3.3.2), операции над базой данных (9.3.5) и операции над модулями БиоАПИ (9.3.6) являются необязательными.

Обобщенные требования соответствия ПБУ каждого подкласса ПБУ приведены в таблице А.1. Подробное описание требований приведено в следующих подразделах. Соответствие относительно опциональных возможностей приведено в А.4.6.

Таблица А.1 — Подтипы соответствия ПБУ

Функция	ПБУ для				
	верификации	идентификации	получения данных	Механизм верификации	Механизм идентификации
Функции управления компонентом					
BioSPI_BSPLoad	X	X	X	X	X
BioSPI_BSPUnload	X	X	X	X	X
BioSPI_BSPAttach	X	X	X	X	X
BioSPI_BSPDetach	X	X	X	X	X
BioSPI_QueryUnits	X	X	X		
BioSPI_QueryBFPs					
BioSPI_ControlUnit			X		
Функции управления дескриптором					
BioSPI_FreeBIRHandle	X	X	X	X	X
BioSPI_GetBIRFromHandle	X	X	X	X	X
BioSPI_GetHeaderFromHandle	X	X	X	X	X
Функции управления обратными вызовами и событиями					
BioSPI_EnableEvents	X	X	X		
BioSPI_SetGUICallbacks					
Биометрические функции					
BioSPI_Capture			X		
BioSPI_CreateTemplate				X	X
BioSPI_Process				X	X
BioSPI_ProcessWithAuxBIR					
BioSPI_VerifyMatch				X	X
BioSPI_IdentifyMatch					X
BioSPI_Enroll	X	X			
BioSPI_Verify	X	X			
BioSPI_Identify		X			
BioSPI_Import					
BioSPI_PresetIdentifyPopulation		X			X

Окончание таблицы А.1

Функция	ПБУ для				
	верификации	идентификации	получения данных	Механизм верификации	Механизм идентификации
Функции управления базой данных					
BioSPI_DbOpen					
BioSPI_DbClose					
BioSPI_DbCreate					
BioSPI_DbDelete					
BioSPI_DbSetMarker					
BioSPI_DbFreeMarker					
BioSPI_DbStoreBIR					
BioSPI_DbGetBIR					
BioSPI_DbGetNextBIR					
BioSPI_DbDeleteBIR					
Функции модуля БиоАПИ					
BioSPI_SetPowerMode					
BioSPI_SetIndicatorStatus					
BioSPI_GetIndicatorStatus					
BioSPI_CalibrateSensor					
Служебные функции					
BioSPI_Cancel	X	X	X	X	X
BioSPI_Free	X	X	X	X	X

А.4.1 Соответствие ПБУ для верификации спецификации БиоАПИ

ПБУ для верификации — ПБУ, способные выполнять сопоставление «один к одному» (верификацию) и не способные выполнять сопоставление «один ко многим» (идентификацию).

Примечание — Сопоставление «один ко многим» может быть реализовано в виде последовательности запросов сопоставления «один к одному».

Для соответствия настоящему стандарту ПБУ для верификации должен поддерживать следующие биометрические функции:

- ***BioSPI_Verify***;
- ***BioSPI_Enroll***.

А.4.1.1 Для биометрической функции ***BioSPI_Verify*** необходима поддержка только лучшего значения MaxFMRRequested. ПБУ должен возвращать это поддерживаемое значение (*FMRAchieved*). Если в ПБУ реализована грубая оценка, то ПБУ должен указать в реестре компонентов это свойство.

Примечание — Критерии использования ОЛС для оценки схожести и пороговой классификации приведены в разделе С.4 приложения С.

Использование параметра *Subtype* является необязательным.

Возвращение полезной информации (*Payload*) необходимо только в том случае, если она связана со входным контрольным шаблоном и если оценка схожести превышает значение *FMRAchieved* (ПБУ должен внести в реестр компонентов минимальное значение ОЛС, требуемое для возвращения полезной информации).

Возвращение *AdaptedBIR* является необязательным. Возвращение исходных данных (*AuditData*) является необязательным.

ПБУ должны предоставлять весь необходимый интерфейс пользователя (в качестве интерфейса пользователя по умолчанию), связанный с получением данных, являющимся частью операции верификации. Поддержка контролируемого приложением ГИП является необязательной.

А.4.1.2 Для биометрической функции ***BioSPI_Enroll*** должно приниматься только назначение BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY (регистрация только для верификации). Если запрашивается другое неподдерживаемое назначение, то BioAPI_RETURN должно содержать тип ошибки.

Использование *Payload* (возвращение полезной информации) является необязательным (то есть, если полезная информация предусмотрена приложением, но не поддерживается ПБУ, она не обязательно должна быть принята).

Принятие и использование параметра *Subtype* является необязательным.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), он должен принимать входной параметр *OutputFormat* и возвращать *NewTemplate* в этом формате.

Использование входного *ReferenceTemplate* (если он предоставлен) для создания выходного *NewTemplate* является необязательным.

Возврат исходных данных (*AuditData*) является необязательным.

Все ПБУ должны предоставлять весь необходимый интерфейс пользователя (в качестве интерфейса пользователя по умолчанию), связанный с получением данных, являющимся частью операции *Enroll*. Поддержка контролируемого приложением ГИП является необязательной.

А.4.2 Соответствие ПБУ для идентификации спецификации БиоАПИ

ПБУ для идентификации — ПБУ, способные выполнять как сопоставление «один ко многим» (идентификацию), так и сопоставление «один к одному» (верификацию). Для соответствия стандарту БиоАПИ ПБУ для идентификации должен поддерживать следующие биометрические функции:

- ***BioSPI_Verify***;
- ***BioSPI_Identify***;
- ***BioSPI_Enroll***.

А.4.2.1 Для биометрической функции ***BioSPI_Verify*** необходима поддержка только лучшего значения *MaxFMRRequested*. ПБУ должен возвращать это поддерживаемое значение (*FMRAchieved*). Если в ПБУ реализована грубая оценка, то ПБУ должен указать в реестре компонентов это свойство.

Примечание — Критерии использования ОЛС для оценки схожести и пороговой классификации приведены в разделе С.4 приложения С.

Принятие и использование параметра *Subtype* является необязательным.

Возвращение полезной информации (*Payload*) необходимо только в том случае, если она связана со входным *ReferenceTemplate* и если оценка схожести превышает значение *FMRAchieved*. ПБУ должен внести в реестр компонентов минимальное значение ОЛС, требуемое для возвращения полезной информации.

Возвращение *AdaptedBIR* является необязательным. Возвращение исходных данных (*AuditData*) является необязательным.

Все ПБУ должны предоставлять весь необходимый ГИП (в качестве интерфейса пользователя по умолчанию), связанный с получением данных, являющийся частью операции верификации. Поддержка контролируемого приложением ГИП является необязательной.

А.4.2.2 Для биометрической функции ***BioSPI_Identify*** необходима поддержка только лучшего значения *MaxFMRRequested*. Должен быть обеспечен возврат списка *Candidates*. ПБУ может возвратить значения поля *FMRAchieved* в виде следующего ближайшего шага/приращения. ПБУ должен указать в реестре компонентов, реализована ли в нем грубая оценка.

Примечание — Критерии использования ОЛС для оценки схожести и пороговой классификации приведены в разделе С.4 приложения С.

Поддержка режима категоризации является необязательной.

Возврат исходных данных (*AuditData*) является необязательным.

Все ПБУ должны предоставлять весь необходимый ГИП (в качестве интерфейса пользователя по умолчанию), связанный с получением данных, являющийся частью операции ***Identify.*** Поддержка контролируемого приложением ГИП является необязательной.

А.4.2.3 Для биометрической функции ***BioSPI_Enroll*** должны приниматься только следующие значения:

- BioAPI_PURPOSE_ENROLL;
- BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY;
- BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY.

Если запрашивается другое неподдерживаемое назначение, то BioAPI_RETURN должно содержать тип ошибки. Принятие *Payload* является необязательным.

Принятие и использование параметра *Subtype* является необязательным.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), то он должен принимать входной параметр *OutputFormat* и возвращать *NewTemplate* в этом формате.

Использование входного *ReferenceTemplate* (если он предоставлен) для создания выходного *NewTemplate* является необязательным.

Возврат исходных данных (*AuditData*) является необязательным.

ПБУ должны предоставлять весь необходимый ГИП (в качестве интерфейса пользователя по умолчанию), связанный с получением данных, являющийся частью операции регистрации. Поддержка контролируемого приложением ГИП является необязательной.

А.4.3 Соответствие ПБУ для получения данных спецификации БиоАПИ

ПБУ для получения данных — ПБУ, предоставляющие интерфейс к одному или нескольким биометрическим датчикам и возвращающие промежуточные ЗБИ, которые могут затем использоваться другими ПБУ (например, биометрическим механизмом БиоАПИ, см. А.4.4 и А.4.5). Данный тип ПБУ не предоставляет возможности для обработки или сопоставления получаемых данных. Для соответствия стандарту БиоАПИ ПБУ для получения данных должно поддерживать следующую биометрическую функцию:

- BioSPI_Capture.

А.4.3.1 Для биометрической функции ***BioSPI_Capture*** возврат исходных данных (*AuditData*) является необязательным.

Принятие и использование параметра *Subtype* является необязательным.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), то он должен принимать входной параметр *OutputFormat* и возвращать *NewTemplate* в этом формате.

ПБУ должны предоставлять весь необходимый ГИП (в качестве интерфейса пользователя по умолчанию), связанный с операцией *BioSPI_Capture*. Поддержка контролируемого приложением ГИП является необязательной.

А.4.4 Соответствие механизма верификации спецификации БиоАПИ

Механизмы биометрической верификации — ПБУ, содержащие алгоритмы биометрической обработки и сопоставления «один к одному», но не выполняющие получение биометрических данных. Как правило, они используются совместно с другим ПБУ, управляющим операцией получения данных (устройство получения данных либо полноценный ПБУ). Для соответствия настоящему стандарту биометрический механизм должен поддерживать следующие биометрические функции:

- BioSPI_CreateTemplate;

- BioSPI_Process;

- BioSPI_VerifyMatch.

А.4.4.1 Для биометрической функции ***BioSPI_CreateTemplate*** должно приниматься только назначение BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY (регистрация только для верификации). Если установлено другое назначение, то BioAPI_RETURN должно содержать тип ошибки.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), то он должен принимать входной параметр *OutputFormat* и возвращать *NewTemplate* в этом формате.

Принятие полезной информации (*Payload*) является необязательным. Обновление шаблона (с использованием входного *ReferenceTemplate*) является необязательным.

А.4.4.2 Для биометрической функции ***BioSPI_Process*** должны приниматься только входные *CapturedBIR* с назначением BioAPI_PURPOSE_VERIFY. Если установлено другое назначение, то BioAPI_RETURN должно содержать тип ошибки.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), то он должен принимать входной параметр *OutputFormat* и возвращать *NewTemplate* в этом формате.

А.4.4.3 Для биометрической функции ***BioSPI_VerifyMatch*** должны приниматься только входные *ProcessedBIR* с назначением BioAPI_PURPOSE_VERIFY и входные *ReferenceTemplate* с назначением BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY. Если установлены другие назначения, то BioAPI_RETURN должно содержать тип ошибки.

Необходима поддержка только лучшего значения *MaxFMRRequested*, однако ПБУ должен возвращать это поддерживаемое значение (*FMRAchieved*). Если в ПБУ реализована грубая оценка, то ПБУ должен указать в реестре компонентов это свойство.

Примечание — Критерии использования ОЛС для оценки схожести и пороговой классификации приведены в разделе С.4 приложения С.

Возвращение полезной информации (*Payload*) необходимо только в том случае, если она связана со входным контрольным шаблоном и если оценка схожести превышает значение *FMRAchieved* (ПБУ должен внести в реестр компонентов минимальное значение ОЛС, необходимое для возвращения полезной информации).

Возвращение *AdaptedBIR* является необязательным.

А.4.5 Соответствие механизма идентификации спецификации БиоАПИ

Для соответствия настоящему стандарту механизм биометрической идентификации должен удовлетворять всем требованиям соответствия, предъявляемым к механизму биометрической верификации, за исключением того, что функция ***BioSPI_CreateTemplate*** должна принимать *CapturedBIR* со значением параметра *Purpose* BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY и функция ***BioSPI_Process*** должна принимать *CapturedBIR* со значением параметра *Purpose* BioAPI_PURPOSE_IDENTIFY; также должно быть обеспечено выполнение следующей биометрической функции:

- BioSPI_IdentifyMatch.

А.4.5.1 Для биометрической функции ***BioSPI_IdentifyMatch*** необходима поддержка только ближайшего лучшего значения *MaxFMRRequested*. Требуется возврат списка *Candidates*, однако ПБУ может возвратить значения

для поля *FMRAchieved* в виде следующего ближайшего шага/приращения. ПБУ должен указать в реестре компонентов, реализована ли в нем грубая оценка.

Примечание — Критерии использования ОЛС для оценки схожести и пороговой классификации приведены в разделе С.4 приложения С.

Поддержка режима категоризации является необязательной.

А.4.6 Дополнительные возможности

Представленные в данном подразделе возможности являются необязательными для поддержки ПБУ и соответствия БиоАПИ. Однако дополнительные возможности должны соответствовать следующим требованиям:

- реализованные дополнительные возможности должны соответствовать требованиям настоящего стандарта;

- ПБУ должны вносить (при установке) в реестр компонентов (с помощью элемента *OptionsMask*, относящегося к *BSPSchema* в функции реестра ***BioAPI_Util_InstallBSP***, см. 10.2.1) информацию о всех поддерживаемых и неподдерживаемых возможностях;

- документация ПБУ должна включать в себя таблицу, в которой должны быть указаны все поддерживаемые и неподдерживаемые возможности.

А.4.6.1 Дополнительные функции

А.4.6.1.1 Элементарные функции

Несмотря на то что данные функции являются необязательными, рекомендуется, чтобы следующие элементарные функции поддерживались всеми ПБУ, если их выполнение поддерживается базовой технологией.

А.4.6.1.1.1 Если поддерживается функция ***BioSPI_Capture,*** то для ПБУ для верификации необходимо только значение параметра *Purpise* BioAPI_PURPOSE_VERIFY или BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY. Если установлено другое неподдерживаемое назначение, то BioAPI_RETURN должно содержать условие ошибки. Данная функция должна возвратить *CapturedBIR* с назначением BioAPI_PURPOSE_VERIFY, BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY или BioAPI_PURPOSE_ENROLL соответственно.

Если данная функция поддерживается, то ПБУ для идентификации должны принимать все назначения (кроме BioAPI_PURPOSE_AUDIT), даже если нет различий в содержании или формате возвращаемых данных.

Принятие и использование параметра *Subtype* является необязательным.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), он должен принимать входной параметр *OutputFormat* и возвращать *NewTemplate* в этом формате.

Возврат исходных данных (*AuditData*) является необязательным.

Все ПБУ должны предоставлять весь необходимый ГИП (в качестве интерфейса пользователя по умолчанию), связанный с операцией ***BioSPI_Capture.*** Поддержка контролируемого приложением ГИП является необязательной.

А.4.6.1.1.2 Если поддерживается функция ***BioSPI_CreateTemplate,*** то для ПБУ для верификации необходимы только входные *CapturedBIRs* со значением параметра *Purpose* BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY. Если установлено неподдерживаемое назначение, то BioAPI_RETURN должно содержать условие ошибки.

Если данная функция поддерживается, ПБУ идентификации должны принимать входные *CapturedBIRs*, имеющие назначение, связанное с регистрацией.

Принятие полезной информации (*Payload*) является необязательным. Обновление шаблона (входного *ReferenceTemplate*) необязательно.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), он должен принимать входной параметр *OutputFormat* и возвращать *NewTemplate* в этом формате.

А.4.6.1.1.3 Если поддерживается функция ***BioSPI_Process,*** для ПБУ для верификации требуются входные *CapturedBIRs* только со значением параметра *Purpose* BioAPI_PURPOSE_VERIFY. Если установлено неподдерживаемое назначение, то BioAPI_RETURN должно содержать условие ошибки.

Если данная функция поддерживается, ПБУ для идентификации должны принимать *CapturedBIRs* со значением параметра *Purpose* или BioAPI_PURPOSE_VERIFY, или BioAPI_PURPOSE_IDENTIFY, даже если нет различий в содержании или формате возвращаемых данных.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), он должен принимать входной параметр *OutputFormat* и возвращать *ProcessedBIR* в этом формате.

А.4.6.1.1.4 Если поддерживается функция ***BioSPI_ProcessWithAuxBIR,*** то для ПБУ для верификации необходимы входные *CapturedBIRs* только со значением параметра *Purpose* BioAPI_PURPOSE_VERIFY. Если установлено неподдерживаемое назначение, то BioAPI_RETURN должно содержать тип ошибки.

Если данная функция поддерживается, то ПБУ для идентификации должны принимать *CapturedBIRs* со значением параметра *Purpose* или BioAPI_PURPOSE_VERIFY, или BioAPI_PURPOSE_IDENTIFY, даже если нет различий в содержании или формате возвращаемых данных.

ПБУ, поддерживающие данную функцию, должны содержать в своей документации формат и содержание (или источник) *AuxiliaryData*, которые принимаются как входной параметр.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), он должен принимать входной параметр *OutputFormat* и возвращать *ProcessedBIR* в этом формате.

А.4.6.1.1.5 Если поддерживается функция ***BioSPI_VerifyMatch,*** то должны приниматься входные *ProcessedBIR* только со значением параметра *Purpose* BioAPI_PURPOSE_VERIFY и входные *ReferenceTemplate* со значением параметра *Purpose* или BioAPI_PURPOSE_ENROLL, или BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY. Если установлены неподдерживаемые назначения, то BioAPI_RETURN должно содержать условие ошибки.

Необходима поддержка только лучшего значения *MaxFMRRequested,* однако ПБУ должен возвращать это поддерживаемое значение (*FMRAchieved*). Если в ПБУ реализована грубая оценка, то ПБУ должен указать в реестре компонентов это свойство.

Примечание — Критерии использования ОЛС для оценки схожести и пороговой классификации приведены в разделе С.4 приложения С.

Возвращение полезной информации (*Payload*) необходимо только в том случае, если она связана со входным контрольным шаблоном и если оценка схожести превышает значение *FMRAchieved.* ПБУ должен внести в реестр компонентов минимальное значение ОЛС, необходимое для возвращения полезной информации.

Возвращение параметра *AdaptedBIR* является необязательным.

А.4.6.1.1.6 Если поддерживается функция ***BioSPI_IdentifyMatch,*** то должны приниматься входные ЗБИ (*ProcessedBIR*) только со значением параметра *Purpose* BioAPI_PURPOSE_ IDENTIFY. Если установлено неподдерживаемое назначение, то BioAPI_RETURN должно содержать условие ошибки.

Необходима поддержка только лучшего значения *MaxFMRRequested.* Требуется возврат списка *Candidates,* однако ПБУ может возвратить значение для поля *FMRAchieved* в виде следующего ближайшего шага/приращения. Если в ПБУ реализована грубая оценка, то ПБУ должен указать в реестре компонентов это свойство.

Примечание — Критерии использования ОЛС для оценки схожести и пороговой классификации приведены в разделе С.4 приложения С.

Поддержка режима категоризации является необязательной.

А.4.6.1.2 Для операций над базой данных не требуется, чтобы ПБУ предоставлял внутреннюю или управляемую базу данных ЗБИ. Однако если она предоставляется, то должны предоставляться все функции управления для доступа и обслуживания базы данных. Все предоставляемые функции управления базой данных должны соответствоать приведенным описаниям.

А.4.6.1.3 Функция ***BioSPI_Import*** не является обязательной. Если она предоставляется, то это должно быть отображено в реестре компонентов и функция должна быть реализована в соответствии с приведенным описанием.

ПБУ для верификации должен выполнять импортированные только тех данных, у которых параметр назначения установлен в BioAPI_PURPOSE_ENROLL или BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY. ПБУ для идентификации должен выполнять импортирование только тех данных, у которых параметр назначения установлен в BioAPI_PURPOSE_ENROLL или BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY.

Если ПБУ поддерживает более одного формата данных ЗБИ (как указано в схеме ПБУ в реестре компонентов), то он должен принимать входной параметр *OutputFormat* и возвращать *ConstructedBIR* в указанном формате.

А.4.6.1.4 Функция ***BioSPI_PresetIdentifyPopulation*** не является обязательной. Если она предоставляется, то это должно быть отображено в реестре компонентов и функция должна быть реализована в соответствии с приведенным описанием.

А.4.6.1.5 Операции с модулями БиоАПИ не являются обязательными, так же как и все операции над модулями BioAPI. Если они предоставляются, то это должно быть отображено в реестре компонентов и операции должны быть реализованы в соответствии с приведенным описанием.

А.4.6.2 Дополнительные подфункции

В таблице А.2 приведены дополнительные возможности, для каждой из которых ПБУ должен указать в матрице опций в пределах схемы ПБУ реестра компонентов и в документации ПБУ, поддерживает он ее или нет.

Таблица А.2 — Дополнительные возможности

Возможность	Поддерживается	Не поддерживается
Возврат исходных/контрольных данных		
Возврат значения качества		
Управляемый приложением ГИП		
Обратные вызовы потокового ГИП		
Обнаружение источника		
Полезная информация		

Окончание таблицы А.2

Возможность	Поддерживается	Не поддерживается
Подписывание ЗБИ		
Шифрование ЗБИ		
Обновление шаблона		
Адаптация		
Категоризация		
Независимое устройство		
Сопоставление на карте		
Подтипы для получения данных		
ПБФ датчика		
ПБФ архива		
ПБФ сопоставления		
ПБФ обработки		
Грубая оценка схожести		

А.4.6.2.1 Возврат исходных данных

Функции, выполняющие получение биометрических данных от датчика, могут дополнительно поддерживать возврат исходных данных для визуализации или контроля. Если данная функция поддерживает возврат исходных данных, то выходной параметр *AuditData* будет содержать указатель на эти данные. Если возврат исходных данных не поддерживается, то ПБУ возвращает значение BioAPI_UNSUPPORTED_BIR_HANDLE.

А.4.6.2.2 Возврат значения качества

После получения биометрических данных от датчика ПБУ может вычислить значение относительного качества для полученных биометрических данных, которое будет включено в заголовок возвращаемого *CapturedBIR* (и необязательного *AuditData*). Если функция возврата значения качества поддерживается, то поле заголовка будет содержать положительное значение от 1 до 100. Если функция возврата значения качества не поддерживается, то значение поля будет установлено на минус 2. Значение устанавливается в процессе выполнения ***BioSPI_Capture*** и ***BioSPI_Enroll.***

Аналогично для обработанной ЗБИ может быть вычислено другое значение качества, которое добавляется в заголовок *ProcessedBIR* (и необязательного *AdaptedBIR*). Это происходит в процессе выполнения функций ***BioSPI_CreateTemplate, BioSPI_Process, BioSPI_ProcessWithAuxBIR, BioSPI_Verify, BioSPI_VerifyMatch, BioSPI_Enroll*** и ***BioSPI_Import*** (*ConstructedBIR*).

ПБУ должен внести в реестр компонентов (*OptionsMask*) информацию о том, поддерживает он или нет вычисление качества для ЗБИ каждого типа: исходной, промежуточной и обработанной.

Примечание — Дополнительная информация относительно качества данных приведена в 7.49.

А.4.6.2.3 Управляемый приложением ГИП

ПБУ поддерживает необходимые обратные вызовы, чтобы дать возможность приложению управлять видом и поведением ГИП.

Для соответствия настоящему стандарту все ПБУ должны по умолчанию предоставлять возможность визуализации информации интерфейса пользователя (выполнение визуализации необходимо и требуется биометрической технологией) во время операций получения данных. Все предоставляемые ПБУ ГИП должны поддерживать механизм прерывания и отмены оператором.

ПБУ может дополнительно также поддерживать управляемый приложением ГИП. В данном случае ПБУ должен поддерживать функции ***BioSPI_SetGUICallbacks.***

Если поддерживается управляемый приложением ГИП, то ПБУ может дополнительно предоставлять потоковые данные (например, для распознавания по голосу и по лицу). Если потоковые данные предоставляются, то ПБУ должен поддерживать входные параметры *GuiStreamingCallback* и *GuiStreamingCallbackCtx*.

Все ПБУ должны внести в реестр компонентов информацию о поддерживаемых функциях и опциях ГИП.

Примечание — Дополнительная информация относительно вопросов интерфейса пользователя приведена в разделе С.5 приложения С.

А.4.6.2.4 Обратные вызовы потокового ГИП

ПБУ предоставляет потоковые данные ГИП и поддерживает потоковый обратный вызов ГИП.

А.4.6.2.5 Обнаружение источника

ПБУ может обнаруживать наличие доступных образцов и поддерживает событие наличия источника.

Примечание — Примером доступных образцов может служить физический контакт пальца с рабочей поверхностью сканера отпечатков пальцев.

4.6.2.6 Полезная информация

Некоторые ПБУ могут дополнительно поддерживать хранение полезной информации при создании контрольного шаблона и выполнять возврат полезной информации после успешной верификации. Содержание полезной информации не ограничено и может включать в себя секретные данные, такие как личный идентификационный номер или секретные ключи, связанные с пользователем, биометрические данные которого содержатся в ЗБИ. Данные полезной информации создаются в процессе выполнения операций ***BioSPI_Enroll*** или ***BioSPI_CreateTemplate*** и возвращаются как результат операций ***BioSPI_Verify*** или ***BioSPI_VerifyMatch.*** Если функция предусмотрена, то ПБУ должен внести в реестр компонентов максимальный размер поля для полезной информации. Если максимальный размер равен нулю, то это указывает на то, что перенос полезной информации не поддерживается. Если размер входной полезной информации превышает установленный размер, то функция должна возвратить код ошибки. Если перенос полезной информации не поддерживается, выходной параметр *Payload* должен иметь значение минус 1.

Примечание — Дополнительная информация относительно полезной информации приведена в разделе С.5 приложения С.

А.4.6.2.7 Функции безопасности

БиоАПИ поддерживает, но не требует реализации функций безопасности. К тому же он не определяет функции, выходящие за рамки ПИП.

ПБУ вносит в реестр компонентов информацию о возможности поддержки шифрования ЗБИ и создания электронной подписи ЗБИ.

Примечание — Управление ключами, связанное с данными возможностями, не адресуется данным ПИП.

А.4.6.2.8 Обновление шаблона

Функции ***CreateTemplate*** и ***Enroll*** могут быть предоставлены приложением с ранее зарегистрированным шаблоном (*ReferenceTemplate*) для обновления (то есть, ПБУ может использовать этот шаблон отдельно с вновь полученными данными для создания *NewTemplate*). ПБУ должен в любом случае возвратить *NewTemplate* (за исключением события ошибки), но если обновление шаблона не поддерживается, в расчетах может игнорироваться предоставляемый *ReferenceTemplate*.

А.4.6.2.9 Адаптация шаблона

Некоторые ПБУ могут дополнительно обеспечивать возможность использования вновь полученных биометрических данных для обновления контрольной ЗБИ. Данная операция может быть выполнена только в результате успешного завершения функций ***BioSPI_Verify*** или ***BioSPI_VerifyMatch*** (то есть в том случае, если Result равен BioAPI_TRUE). Данная процедура используется для того, чтобы сохранить новую зарегистрированную ЗБИ с самым высоким качеством. ПБУ принимает решение о необходимости выполнения адаптации, основываясь на таких факторах, как качество, прошедшее время и наличие существенных различий. Если ПБУ не поддерживает адаптацию, возвращаемый параметр *AdaptedBIR* должен иметь значение минус 2 (BioAPI_UNSUPPORTED_BIR_HANDLE). Если ПБУ поддерживает адаптацию, но по некоторым причинам не способен выполнить адаптацию или принял решение не выполнять адаптацию, то возвращаемый параметр AdaptedBIR должен иметь значение минус 1 (BioAPI_INVALID_BIR_HANDLE). ПБУ должен внести в реестр компонентов (*OptionsMask*) информацию о поддержке функций адаптации шаблона.

А.4.6.2.10 Категоризация

ПБУ для идентификации могут дополнительно поддерживать методы ограничения совокупности записей базы данных, по которым должен производиться поиск, для снижения времени отклика. Данная функция используется только в ПБУ для идентификации и выполняется только в процессе выполнения операций ***BioSPI_Identify*** и ***BioSPI_IdentifyMatch.*** ПБУ для идентификации должны внести в реестр компонентов информацию о поддержке данной функции. ПБУ для идентификации не поддерживают данную функцию и могут игнорировать входной параметр *Binning* включения/выключения данного режима.

Примечание — Для настройки или модификации стратегии режима категоризации не предоставляется явная поддержка ПИП, кроме опции включения/выключения данного режима.

А.4.6.2.11 Независимое устройство

Поддерживаемое устройство является независимым, если помимо функции получения биометрических данных оно может выполнять как минимум функцию верификации или содержать базу данных ЗБИ.

Примечание — Независимое устройство может содержать два и более модулей БиоАПИ, то есть модуль датчика плюс еще один модуль другой категории БиоАПИ.

А.4.6.2.12 Сопоставление на карте

ПБУ поддерживает возможность выполнения операции сопоставления на смарт-карте. В зависимости от подкласса ПБУ требуется, чтобы ПБУ мог создавать и использовать пригодные для СНК ЗБИ. Может потребоваться также, чтобы ПБУ мог взаимодействовать непосредственно или косвенно с картой.

А.4.6.2.13 Подтип для получения данных

ПБУ поддерживает возможность определения подтипа биометрических данных (например, правый указательный палец или левый глаз). Если данная опция поддерживается и приложение определяет подтип получаемых данных, то ПБУ должен получить данные этих признаков и указать их в заголовке возвращаемого *CapturedBIR*. Это применимо ко всем функциям, требующим получения данных (то есть ***BioSPI _Capture***, ***BioSPI_Enroll***, ***BioSPI_Verify*** или ***BioSPI_Identify***).

А.4.6.2.14 ПБФ датчика

ПБУ поддерживает присоединение альтернативных устройств датчика с помощью связанного ИПФ. Если датчик присоединен, ПБУ использует это устройство для всех функций, требующих получения биометрических данных (то есть ***BioSPI_Capture***, ***BioSPI_Enroll***, ***BioSPI_Verify*** или ***BioSPI_Identify***).

А.4.6.2.15 ПБФ архива

В дополнение к внутренним возможностям работы с архивом ПБУ поддерживает присоединение альтернативного модуля архива. В данном случае ПБУ поддерживает интерфейс к такому модулю через связанный ИПФ. При этом также требуется, чтобы ПБУ поддерживал операции над базой данных BioSPI. Если архив присоединен, ПБУ использует этот архив путем передачи УУИД базы данных, связанной с этим архивом (базой данных), любой функции базы данных.

А.4.6.2.16 ПБФ сопоставления

ПБУ поддерживает присоединение альтернативного модуля (алгоритма) сопоставления. Если модуль присоединен, ПБУ использует альтернативный модуль для любой операции сопоставления, для которой он пригоден (то есть в случае модуля сопоставления «один к одному» может выполняться операция верификации).

А.4.6.2.17 ПБФ обработки

ПБУ поддерживает присоединение альтернативного модуля (алгоритма) обработки. Если модуль присоединен, ПБУ использует альтернативный модуль для всех вызовов функций ***BioSPI_Process*** или ***BioSPI_CreateTemplate***.

А.4.6.2.18 Грубая оценка схожести

ПБУ поддерживает возврат оценки схожести на шаге, на котором она была специально завышена для подсчета попыток атак (см. приложение С, подраздел С.4.6). До тех пор, пока данная опция включена в *OptionsMask*, ПБУ предоставляет конечную квантованную оценку схожести.

Приложение В
(обязательное)

Формат постоянного клиента спецификации ЕСФОБД: формат постоянного клиента БиоАПИ

В.1 Наименование постоянного клиента

ИСО/МЭК СТК1/ПК 37.

В.2 Спецификатор постоянного клиента

257 (0x0101). Данный номер был выделен регистрационной комиссией по ИСО/МЭК 19785-2.

В.3 Наименование формата постоянного клиента

ИСО/МЭК СТК1/ПК 37 Постоянный клиент БиоАПИ.

В.4 Идентификатор формата постоянного клиента

6 (0x0006). Данный номер был зарегистрирован в соответствии с требованиями ИСО/МЭК 19785-2.

В.5 Идентификатор объекта АСН.1 для данного формата постоянного клиента

{iso registration-authority cbeff(19785) organization(0) 257 bir(1) patron-format (6)}

или в обозначении XML

1.1.19785.0.257.1.6

В.6 Область применения

Приложения и ПБУ, требующие ЗБИ, которые полностью поддерживаются структурой данных Си, определенной в настоящем стандарте.

В.7 Версия идентификатора

Данная спецификация формата постоянного клиента имеет версию идентификатора 0x20 (номер редакции издания — 2, номер поправки или изменения данной редакции — 0).

В.8 Версия ЕСФОБД

Настоящий стандарт соответствует ЕСФОБД следующей версии: номер редакции издания — 2, номер поправки или изменения данной редакции — 0.

В.9 Основные сведения

Данный формат постоянного клиента соответствует обязательному стандарту на структуру данных Си для элементов данных ЕСФОБД, которые приведены в разделе 7.

Настоящий формат постоянного клиента предназначен для надежного сохранения или для передачи ЗБИ БиоАПИ как модуля между системами.

Примечание — Пример текста программы для создания формата постоянного владельца БиоАПИ из BioAPI_BIR структуры данных Си в соответствии с 7.4 приведен в приложении D.

Последовательность байтов/битов в представлении данного формата должна быть в формате обратного порядка (big-endian) без каких-либо вставок между элементами данных для всей памяти и передачи между системами.

В.10 Спецификация

Определенные ЕСФОБД элементы данных, не включенные в таблицу В.1, определены в настоящем формате постоянного клиента как абстрактное значение типа NO VALUE AVAILABLE (данные недоступны).

Таблица В.1

Наименование поля БиоАПИ, ссылки БиоАПИ и поля формата постоянного клиента	Наименование элемента данных ЕСФОБД	Длина, байт	Абстрактное значение	Кодировка[a]	Ссылка ЕСФОБД
Длина ЗБИ[b]	Недоступно	4[c]	Длина в байтах полного кода ЗБИ, за исключением поля длины	Целое	Недоступна
BioAPI_VERSION (БиоАПИ, 7.57)	CBEFF_patron_header_version	1	Основной номер равен 2 Дополнительный номер равен 0	'20'	6.5.24

Продолжение таблицы В.1

Наименование поля БиоАПИ, ссылки БиоАПИ и поля формата постоянного клиента	Наименование элемента данных ЕСФОБД	Длина, байт	Абстрактное значение	Кодировка[A]	Ссылка ЕСФОБД
BioAPI_BIR_DATA_TYPE (БиоАПИ, 7.9)	CBEFF_BDB_encryption_options	1	NOT ENCRYPTED ENCRYPTED	'0_' '1_'	6.5.3
	CBEFF_BIR_integrity_options		NOT SIGNED SIGNED	'0_' '2_'	6.5.4
	CBEFF_BDB_processed_level		NO VALUE AVAILABLE RAW INTERMEDIATE PROCESSED	'_0' '_1' '_2' '_4'	6.5.11
(признак индекса)	Недоступно		INDEX PRESENT	'8_'	Недоступна
BioAPI_BIR_BIOMETRIC_DATA_FORMAT (БиоАПИ, 7.6)	CBEFF_BDB_format_owner CBEFF_BDB_format_type	2 2	FormatOwner FormatType	Целое Целое	6.5.1; 6.5.2
BioAPI_QUALITY (БиоАПИ, 7.49)	CBEFF_BDB_quality	1	NOT SUPPORTED NOT SET (NO VALUE AVAILABLE) VALUE	-2 -1 0 — 100	6.5.15
BioAPI_BIR_PURPOSE (БиоАПИ, 7.12)	CBEFF_BDB_purpose	1	NO VALUE AVAILABLE VERIFY IDENTIFY ENROLL ENROLL_FOR_VERIFICATION_ONLY ENROLL_FOR_IDENTIFICATION_ONLY AUDIT	0 1 2 3 4 5 6	6.5.14
BioAPI_BIR_BIOMETRIC_TYPE (БиоАПИ, 7.8)	CBEFF_BDB_biometric_type	4	NO VALUE AVAILABLE MULTIPLE FACIAL FEATURES VOICE FINGERPRINT IRIS RETINA HAND GEOMETRY SIGNATURE DYNAMICS KEYSTROKE DYNAMICS LIP MOVEMENT THERMAL FACE IMAGE THERMAL HAND IMAGE GAIT OTHER PASSWORD	'00 00 00 00' '00 00 00 01' '00 00 00 02' '00 00 00 04' '00 00 00 08' '00 00 00 10' '00 00 00 20' '00 00 00 40' '00 00 00 80' '00 00 01 00' '00 00 02 00' '00 00 04 00' '00 00 08 00' '00 00 10 00' '40 00 00 00' '80 00 00 00'	6.5.6

Продолжение таблицы В.1

Наименование поля БиоАПИ ссылки БиоАПИ и поля формата постоянного клиента	Наименование элемента данных ЕСФОБД	Длина, байт	Абстрактное значение	Кодировка[a]	Ссылка ЕСФОБД
BioAPI_BIR_BIOMETRIC_PRODUCT_ID (БиоАПИ, 7.7)	CBEFF_BDB_product_owner CBEFF_BDB_product_type	2 2	ProductOwner ProductType NO VALUE AVAILABLE	Целое ненулевое Целое ненулевое '00 00 00 00'	6.5.12 6.5.13
BioAPI_DATE (БиоАПИ, 7.20) (дата создания)	CBEFF_BDB_creation_date	4	NO VALUE AVAILABLE Год, месяц, день	'00 00 00 00' Целое ненулевое	6.5.9
BioAPI_TIME (БиоАПИ, 7.52) (время создания)		3	NO VALUE AVAILABLE Час, минуты, секунды	'99 99 99' Целое	
BioAPI_BIR_SUBTYPE (БиоАПИ, 7.14)	CBEFF_BDB_biometric_subtype	1	NO VALUE AVAILABLE LEFT RIGHT THUMB POINTERFINGER MIDDLEFINGER RINGFINGER LITTLEFINGER MULTIPLE	'00' '01' '02' '04' '08' '10' '20' '40' '80'	6.5.7
BioAPI_DATE (БиоАПИ, 7.20) (дата истечения срока)	CBEFF_BDB_validity_period[d]	4	NO VALUE AVAILABLE Год, месяц, день	00 00 00 00' Целое ненулевое	6.5.16
BioAPI_BIR_SECURITY_BLOCK_ FORMAT (БиоАПИ, 7.13)	CBEFF_SB_format_owner CBEFF_SB_format_type	2 2	SecurityFormatOwner SecurityFormatType NO VALUE AVAILABLE	Ненулевое целое Ненулевое целое '00 00 00 00'	6.5.25 6.5.26
BioAPI_UUID (БиоАПИ, 7.56) (index)	CBEFF_BDB_indexe[e]	16	Индекс базы данных ЗБИ	Целое	6.5.10
Длина ББД[g]	Недоступно	4	Length in bytes of BDB	Целое	Недоступно
BioAPI_ DATA (БиоАПИ, 7.19) (BiometricData)	Блок биометрических данных (ББД)[f]	Непостоянный	BiometricData		6.2.2
Длина SB[g]	Недоступно	4	Длина SB в байтах	Целое	Недоступно
BioAPI_DATA (БиоАПИ, 7.19) (SecurityBlock)	Security Block (SB)[h]	Непостоянный	SecurityBlock		6.2.3

Окончание таблицы В.1

[a] По умолчанию кодировка представляет собой целочисленные значения.

[b] Длина ЗБИ не присутствует в представлении структуры данных Си ЗБИ, но присутствует в форматах сохранения и передачи данных. В данном случае длина — длина всей ЗБИ, за исключением самого поля длины ЗБИ, но содержащая все остальные поля заголовков, поле длины ББД, ББД, поле длины блока безопасности и блок безопасности.

[c] ЗБИ БиоАПИ не может содержать ББД и/или блок безопасности, являющиеся достаточно большими — код целой ЗБИ более 4 байтов.

[d] Данное поле БиоАПИ представляет собой вторую часть элемента данных CBEFF_BDB_validity_period; поле BioAPI_DATE (дата создания) — первую часть.

[e] Поле УУИД/индекс присутствует только в том случае, если установлен соответствующий признак в параметре BIR_DATA_TYPE (все остальные поля заголовка ЗБИ присутствуют всегда).

[f] Формат и содержимое ББД определены спецификацией формата ББД, указываемой с помощью значений владельца формата и типа формата, закодированных в параметре BioAPI_BIR_BIOMETRIC_DATA_FORMAT.

[g] БиоАПИ не поддерживает ББД или блок безопасности, длина которых более 4 байтов. Длина блока безопасности всегда должна указываться. Если блок безопасности не предусмотрен, его длина должна быть установлена на ноль.

[h] Формат и содержимое блока безопасности определены спецификацией формата блока безопасности, указываемой значениями владельца формата безопасности и типом формата безопасности, закодированных в параметре BioAPI_BIR_SECURITY_BLOCK_FORMAT.

Формат хранения ЗБИ представляет собой последовательность бинарных полей фиксированной длины (отдельно от содержания полей ББД и блока безопасности). Последовательность полей приведена в таблице В.1. В данном виде заголовок может быть длиной либо 36, либо 52 байта, в зависимости от присутствия необязательного поля УУИД (см. таблицу В.1, сноска е).

Примечание — Данный формат также используется для обмена данными ЗБИ, однако в процессе обмена данными на системном, прикладном или структурном уровне может произойти изменение ЗБИ в другой формат постоянного клиента ЕСФОБД. Дополнительная информация о преобразовании формата постоянного клиента приведена в ИСО/МЭК 19785; информация о протоколе биометрического межсетевого обмена (ПБО) — в ИСО/МЭК 24708.

В.11 Формулировка соответствия формата постоянного клиента

В.11.1 Идентифицирующая информация

Необходимая информация	Ссылки формата постоянного клиента
Наименование постоянного клиента	См. В.1
Идентификатор постоянного клиента	См. В.2
Наименование формата постоянного клиента	См. В.3
Идентификатор формата постоянного клиента	См. В.4
Идентификатор объекта АСН.1 формата постоянного клиента	См. В.5
Область применения	См. В.6
Версия формата постоянного клиента	См. В.7
Версия ЕСФОБД	См. В.8

В.11.2 Элементы данных и абстрактные значения, определяемые ЕСФОБД

Наименование элемента данных ЕСФОБД	Обязательный/необязательный	Наименование поля формата постоянного клиента (наименование структуры БиоАПИ)	Абстрактное значение определено	Кодирование определено
CBEFF_patron_header_version	Обязательный	HeaderVersion (BioAPI_VERSION)	Да	Да
CBEFF_BDB_encryption_options	Обязательный	Type (BioAPI_BIR_DATA_TYPE_ ENCRYPTED)	Да	Да

Продолжение таблицы

Наименование элемента данных ЕСФОБД	Обязательный/необязательный	Наименование поля формата постоянного клиента (наименование структуры БиоАПИ)	Абстрактное значение определено	Кодирование определено
CBEFF_BIR_integrity_options	Обязательный	Type (BioAPI_BIR_DATA_TYPE_SIGNED)	Да	Да
CBEFF_BDB_processed_level	Обязательный	Type (BioAPI_BIR_DATA_TYPE_RAW, BioAPI_BIR_DATA_TYPE_INTERMEDIATE, BioAPI_BIR_DATA_TYPE_PROCESSED)	Да	Да
CBEFF_BDB_format_owner	Обязательный	FormatOwner (BioAPI_BIR_BIOMETRIC_DATA_FORMAT)	Да	Да
CBEFF_BDB_format_type	Обязательный	FormatType (BioAPI_BIR_BIOMETRIC_DATA_FORMAT)	Да	Да
CBEFF_BDB_quality	Обязательный	Quality (BioAPI_QUALITY)	Да	Да
CBEFF_BDB_purpose	Обязательный	Purpose (BioAPI_BIR_PURPOSE)	Да	Да
CBEFF_BDB_biometric_type	Обязательный	FactorsMask (BioAPI_BIR_BIOMETRIC_TYPE)	Да	Да
CBEFF_BDB_product_owner	Обязательный	ProductOwner (BioAPI_BIR_BIOMETRIC_PRODUCT_ID)	Да	Да
CBEFF_BDB_product_type	Обязательный	ProductType (BioAPI_BIR_BIOMETRIC_PRODUCT_ID)	Да	Да
CBEFF_BDB_creation_date	Обязательный	CreationDTG (BioAPI_DTG)	Да	Да
CBEFF_BDB_biometric_subtype	Обязательный	Subtype (BioAPI_BIR_SUBTYPE)	Да	Да
CBEFF_BDB_validity_period	Обязательный	ExpirationDate (BioAPI_DATE)	Да	Да
CBEFF_SB_format_owner	Обязательный	SecurityFormatOwner BioAPI_BIR_SECURITY_BLOCK_FORMAT	Да	Да
CBEFF_SB_format_type	Обязательный	SecurityFormatType BioAPI_BIR_SECURITY_BLOCK_FORMAT	Да	Да
CBEFF_BDB_index	Необязательный	Index (BioAPI_UUID)	Да	Да

B.11.3 Определенные постоянным клиентом элементы данных и абстрактные значения

Данный формат постоянного клиента определяет дополнительные абстрактные значения или исключает их для двух элементов данных, определенных ЕСФОБД. ИСО/МЭК СТК 1 ПК 37, являющийся постоянным клиентом данного формата постоянного клиента ЕСФОБД, определяет, что такие исключения или различия выполняются в соответствии с утверждением требований ЕСФОБД для соответствия ИСО/МЭК 19785-1. Рассматриваемыми элементами данных являются следующие:

CBEFF_BDB_biometric_type

CBEFF_BDB_biometric_subtype

Приложение С
(справочное)

Краткий обзор стандарта

С.1 ПИП прикладного уровня

С.1.1 Прикладной уровень является верхним уровнем, на котором реализованы основные биометрические функции, обычно используемые приложением для реализации биометрических возможностей.

С.1.2 ПИП прикладного уровня содержит все функции, которые требуются приложению для биометрической верификации. Поэтому, по возможности, количество дополнительных функциональных возможностей сводится к минимуму. Основной дополнительной функцией является функция идентификации; только специализированные ПБУ могут реализовывать данную функцию для больших совокупностей, и основной причиной того, что возможности базы данных вынесены в интерфейс, является обеспечение возможности ПБУ управлять этими большими совокупностями.

С.1.3 БиоАПИ предназначен для использования как приложениями, так и разработчиками ПБУ. Для обеспечения максимально открытой и простой интеграции технологии (расширяя таким образом ее коммерческую жизнеспособность) предпринятый подход направлен на скрывание или инкапсуляцию до возможного предела сложностей и уникальных аспектов индивидуальных биометрических технологий и отдельных реализаций, программных продуктов и устройств изготовителя с обеспечением абстракции высокого уровня, которая может использоваться в ряде потенциальных программных приложений. Данный подход также обеспечивает повышение универсальности интерфейса для обращения к большему набору потенциальных биометрических технологий и приложений. Доступ к биометрическим функциям осуществляется с помощью набора стандартных интерфейсов, определенных в настоящем стандарте, и поставщики биометрической услуги предоставляют данные механизмы с помощью связанных со стандартом интерфейсов. Теоретически ПБУ, снабженные конструкторами, соответствующими настоящему стандарту интерфейса, могут использоваться для достижения, как минимум, основного уровня функциональных возможностей в пределах любого приложения, разработанного в соответствии с настоящим стандартом.

С.1.4 Настоящий стандарт обеспечивает поддержку нескольких методов верификации, используемых как независимо, так и совместно в «многоуровневой» схеме.

С.2 Биометрические технологии

С.2.1 Базовая модель является одинаковой для всех типов биометрических технологий. Сначала должен быть создан первоначальный регистрационный шаблон пользователя. Это достигается путем сбора ряда образцов с помощью любого устройства получения биометрических данных. Из образцов извлекают характерные особенности, и полученные результаты объединяют в шаблон. Процесс создания данного первоначального шаблона называется регистрацией. Алгоритмы, используемые для создания шаблонов, могут быть запатентованными или нет. Первоначальный шаблон сохраняется приложением как контрольный шаблон (также можно обеспечить хранение шаблона с помощью ПБУ в модуле архива БиоАПИ).

С.2.2 Каждый раз, когда пользователь должен быть верифицирован, «живые» образцы получают с датчика, обрабатывают для представления в пригодной для использования форме и сопоставляют с ранее зарегистрированным шаблоном. Такую форму биометрического установления подлинности называют верификацией, так как проводится проверка, является ли пользователь тем, кем он себя называет (то есть проверяется заявляемая идентичность). Биометрические технологии допускают другую форму установления подлинности, которая называется идентификацией. В данной форме пользователь не должен заявлять идентичность; поставщик биометрической услуги сравнивает обработанные образцы пользователя с указанной совокупностью шаблонов и принимает решение, какой из них имеет наибольшую степень схожести. В зависимости от вероятностей сопоставления может быть принято решение об идентичности пользователя и индивидуума, шаблон которого имеет наибольшую степень схожести.

С.2.3 Возможная архитектура реализации технологии БиоАПИ представлена на рисунке С.1. Различные стадии операций верификации и идентификации показаны в области, обозначенной «Поставщик биометрический услуги». Стадии, указанные над данной областью, относятся к элементарным функциям интерфейса верхнего уровня: получение данных, обработка и сопоставление. На рисунке С.1 видно, что ПБУ (или приложение) обладает свободой в размещении своих функциональных возможностей в данных примитивах. Существует еще одна

степень свободы, не показанная на данном рисунке: конструктор может разместить большинство или все функции ПБУ в самом биометрическом датчике. На практике, если устройство содержит базу данных ЗБИ, все функции могут быть выполнены в независимом устройстве.

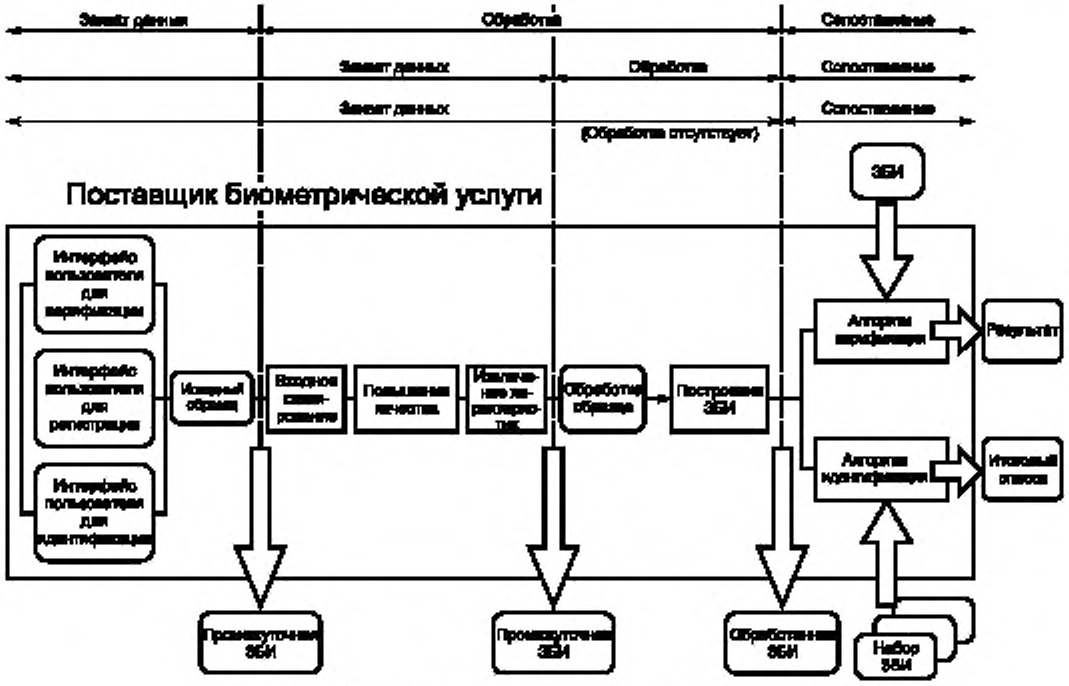


Рисунок С.1 — Возможная архитектура реализации технологии БиоАПИ

С.3 Биометрические функции

С.3.1 В ПИП существует три основные функции абстракции высокого уровня:

1) Регистрация

Образцы поступают с модуля датчика, обрабатываются для представления в пригодной для использования форме, из которой создается шаблон, который возвращается приложению.

2) Верификация

Полученные один или несколько образцов обрабатываются для представления в пригодной для использования форме и сопоставляются с контрольным шаблоном. Результаты сопоставления возвращаются приложению.

3) Идентификация

Полученные один или несколько образцов обрабатываются для представления в пригодной для использования форме и сопоставляются с набором шаблонов. Возвращаемый список показывает, насколько соответствуют образцы основным кандидатам в наборе.

С.3.2 На рисунке С.1 показано, что обработка биометрических данных с момента получения исходных образцов до их сопоставления с шаблоном может включать в себя много этапов. ПИП предоставляет конструктору максимальную возможность свободной расстановки этапов обработки и позволяет разделить этапы обработки между клиентом (к которому присоединено биометрическое устройство) и сервером. Он также допускает возможность использования независимых устройств, которые могут выполнить все этапы биометрической обработки.

С.3.3 Существует ряд причин, по которым обработка и сопоставление должны быть выполнены на сервере:

a) алгоритмы будут выполняться в более защищенной среде (например, в среде с контролируемым доступом, защищенной брандмауэром);

b) клиентская платформа не имеет достаточной производительности для реализации алгоритмов;

c) база данных пользователей и биометрически защищенные ресурсы могут быть размещены на сервере;

d) идентификация по большой совокупности образцов может быть корректно реализована только на сервере.

С.3.4 Элементарные функции

В ПИП есть четыре элементарные функции, которые при последовательном использовании могут обеспечить получение того же результата, что и абстракции высокого уровня. Элементарные функции позволяют приложению определять и контролировать время и место выполнения данных операций компонента, что позволяет им быть разделенными во времени и пространстве.

С.3.4.1 Получение данных

Функция ***BioAPI_Capture*** всегда выполняется на клиентском компьютере; попытка выполнить функцию ***BioAPI_Capture*** на компьютере без биометрического датчика заканчивается возвращением ошибки, так как функция не поддерживается. Должны быть получены один или несколько образцов (для регистрации, верификации или идентификации). Функция ***BioAPI_Capture*** может выполнять полную обработку образца, которая заканчивается созданием ЗБИ для верификации или идентификации. Если обработка не завершена, функция ***BioAPI_Capture*** возвращает промежуточную ЗБИ, указывая, что должна быть вызвана функция ***BioAPI_Process*** (или ***BioAPI_CreateTemplate***). Если обработка завершена, функция ***BioAPI_Capture*** возвращает обработанную ЗБИ, указывая, что функция ***BioAPI_Process*** выполнена. Приложение устанавливает тип и назначение образцов, давая возможность ПБУ выполнить специальную обработку. Данное назначение записывается в заголовке созданной ЗБИ.

С.3.4.2 Обработка

Алгоритмы обработки могут быть доступны как на клиентском компьютере, так и на сервере. Функция *BioAPI_Process* предназначена для выполнения необходимой для верификации или идентификации (но не регистрации) обработки образцов. Функция всегда принимает в качестве входного параметра промежуточную ЗБИ и может выполнять обработку до получения конечного шаблона определенного типа, соответствующего указанному назначению. Если функция обработки выполняется на клиентском компьютере, то она может возвращать как обработанную, так и промежуточную ЗБИ, что указывает на необходимость вызова функции ***BioAPI_Process*** на сервере. Сервер завершает обработку и возвращает обработанную ЗБИ приложению. Приложение может передать обработку на сервер, а может, с целью экономии пропускной способности (и мощности сервера), вызвать функцию ***BioAPI_Process*** на клиентском компьютере.

Примечание — Размеры обработанных ЗБИ всегда меньше промежуточных; уменьшение размера ЗБИ зависит от применяемой технологии, а также от степени обработки, выполненной функцией ***BioAPI_Capture.***

С.3.4.3 Сопоставление — верификация и идентификация

На данном этапе проводится сравнение обработанной ЗБИ с одним шаблоном (верификация) или с набором шаблонов (идентификация). Поддержка функции идентификации является необязательной, но поддерживаемые функции сопоставления всегда доступны на сервере и могут быть доступны на клиентском компьютере.

В.3.4.4 Создание шаблона

Функция ***BioAPI_CreateTemplate*** предназначена для создания шаблона из зарегистрированных биометрических данных. Входным параметром функции всегда является промежуточная ЗБИ, а результатом — шаблон (то есть

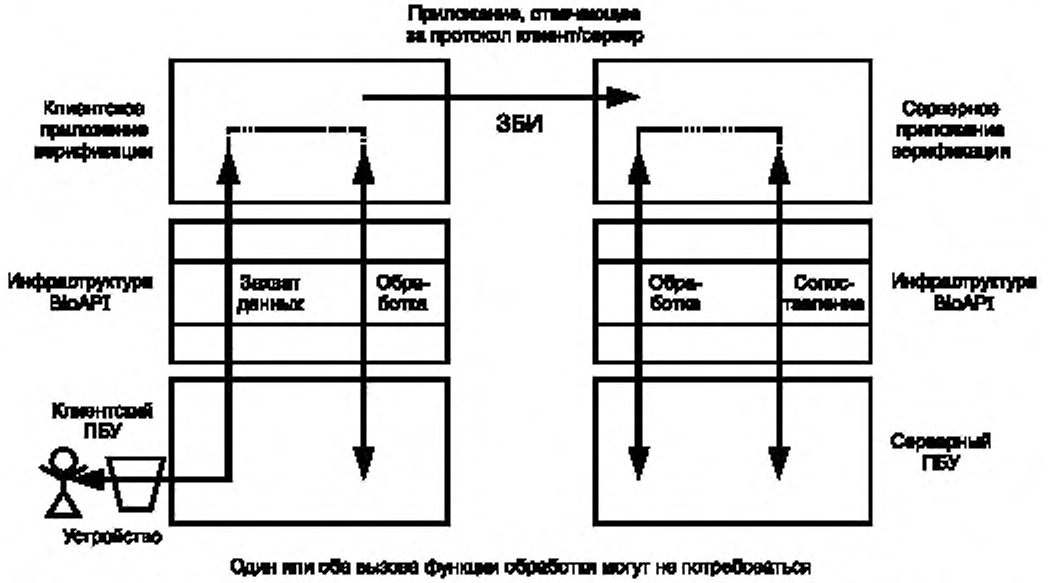


Рисунок С.2 — Пример реализации технологии клиент-сервер с использованием элементарных функций

обработанная ЗБИ с назначением BioAPI_PURPOSE_ENROLL, BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY или BioAPI_PURPOSE_ENROLL_FOR_IDENTIFICATION_ONLY). Дополнительно функция ***BioAPI_CreateTemplate*** может принимать старый шаблон и создавать из него новый шаблон, что является адаптацией или обновлением старого шаблона с использованием новых биометрических образцов в виде промежуточных ЗБИ. ПБУ может дополнительно позволить приложению предоставить для сохранения полезную информацию (С.5).

На рисунке С.2 показано возможное распределение элементарных функций между платформами сервера и клиента, в котором приложение контролирует место и время вызова операции, а также обмен информацией между клиентом и сервером.

Примечание — Реализация технологии клиент — сервер также может быть выполнена с использованием биометрического протокола межсетевого обмена, который будет определен в ИСО/МЭК 24708 [6], для взаимодействия одной инфраструктуры с другой.

С.4 Оценивание схожести и выбор порога в БиоАПИ

С.4.1 Исходные биометрические образцы являются комплексными аналоговыми потоками данных, поступающими от устройств регистрации (модулей датчика). Не существует двух идентичных образцов, полученных от одного пользователя. Шаблоны являются цифровым результатом обработки и сжатия данных образцов и не являются точным представлением пользователя. Поэтому результаты любого сопоставления образцов с контрольным шаблоном могут быть выражены только в терминах вероятности.

С.4.2 Существует два возможных критерия для оценки результатов сопоставления: вероятность ошибки ложного совпадения (ОЛС) и вероятность ошибки ложного отказа (ОЛО). ОЛС — вероятность того, что образец ложно соответствует представленному шаблону, а ОЛО — вероятность того, что образец ложно отклонен (то есть должен соответствовать, но не признан таковым). В зависимости от обстоятельств приложение может быть заинтересовано использовать либо тот, либо другой критерий.

С.4.3 Функции БиоАПИ позволяют приложению запрашивать порог сопоставления в виде максимального значения ОЛС (то есть предельного значения вероятности ложного доступа).

С.4.4 Для каждой технологии и реализации каждое значение ОЛС имеет соответствующее значение ОЛО (существует взаимооднозначная взаимосвязь ОЛО с ОЛС). Определяя либо ОЛС, либо ОЛО, определяют другое значение. Для соответствия требованиям БиоАПИ необходимо, чтобы все технологии и реализации поддерживали пороговую классификацию, основанную на значении ОЛС.

С.4.5 Основным возвращаемым результатом является фактическое значение ОЛС, полученное в результате сопоставления (оценивания схожести). Для функций ***BioAPI_Identify*** и ***BioAPI_IdentifyMatch*** эти результаты содержатся в массиве кандидатов.

С.4.6 Процесс возврата приложению оценок схожести будет защищен слабо, если не были предприняты соответствующие меры. Приложение «мошенника» может совершить нападение в виде «поиска экстремума», последовательно модифицируя образцы случайным образом и сохраняя только те изменения, которые приводят к увеличению возвращаемой оценки схожести. Таким образом может быть создано искусственное изображение, которое может «обмануть» биометрическую систему. Однако данный метод нападения может быть исключен путем разрешения возврата приложению только дискретных приращений оценки схожести. Уровень квантования, который требуется для нейтрализации подобного рода нападений, зависит от типа используемой биометрической характеристики и используемых алгоритмов. Возврат дискретных (грубоквантованных) приращений оценки схожести ПБУ является необязательным; ПБУ может возвращать как непрерывную (конечно квантованную), так и дискретную оценку схожести. Если предоставляется грубая оценка, то это должно быть отображено в маске опций в схеме ПБУ в реестре компонентов.

Примечание 1 — Нападение в виде «поиска экстремума» является изощренным методом и рассчитано на слабость системы безопасности.

Примечание 2 — Грубое квантование неизбежно приводит к снижению информативности оценки схожести. По существу, его использование, в том числе и в оценке схожести, приводит к ухудшению нисходящей мультимодальной или мультибиометрической комбинации.

С.4.7 Использование значений ОЛС для представления оценок схожести предусмотрено для обеспечения нормализации и возможности сравнения различных технологий, а также предоставления общепонятных средств выбора порогов и интерпретации результатов. Не предполагается строгое измерение эксплуатационных характеристик (то есть точное измерение значения ОЛС для отдельного индивидуального образца сопоставления). Кроме того, конструктор ПБУ несет ответственность за точное отображение внутренней структуры оценки схожести, основанной на рабочей характеристике получателя (РХС), в значение ОЛС.

С.4.8 Может быть применен другой способ оценки нормализации. Значения оценки схожести и порога принятия решения могут быть определены в произвольном масштабе, зависящем от биометрической модальности и выбора изготовителя оборудования. К тому же оценка схожести может быть определена в одном из нескольких нормализованных масштабов при условии, что технология и реализация будут сопоставимы. В зависимости от обстоятельств одному приложению может быть предпочтительнее использовать один масштаб оценки схожести, другому приложению — другой. Таким образом, нормированный масштаб является масштабом ошибочного соответствия, используемым в БиоАПИ, в котором порог принятия решения для отдельного значения

является вероятностью ошибки ложного соответствия в числовом виде, равной пороговому значению в этом масштабе.

Используют следующие шкалы оценки схожести:

a) исходная шкала оценки (определена для разработчиков и изготовителей). Предоставляет улучшенную поддержку и упрощенную модель нормализованной оценки с использованием функции плотности вероятности (ФПВ), полученной при исследовании приложения с целью улучшения преобразования;

b) логарифмическая шкала ОЛС (логарифмическая шкала ошибочного соответствия). Предоставляет значительный динамический интервал;

c) логарифмическая шкала по основанию 10 (логарифм от отношения коэффициентов вероятности истинного к ложному). Обеспечивает оптимальное приближение к комбинации двух независимых биометрических характеристик в соответствии со статистической моделью теории согласования.

Некоторые из этих альтернативных мер применимы в мультимодальных реализациях. Настоящий стандарт не рассматривает поддержку таких альтернативных шкал оценки, которые, вероятно, будут использоваться в будущем.

С.5 Полезная информация

С.5.1 Не существует двух идентичных образцов, полученных от одного пользователя. Поэтому невозможно использовать непосредственно биометрические образцы в качестве криптографических ключей. В соответствии с БиоАПИ шаблон может быть взаимосвязан с криптографическим ключом, который может предоставляться после успешной верификации.

С.5.2 Полезная информация предоставляется приложением в процессе выполнения операции Enroll или CreateTemplate. ПБУ несет ответственность за хранение полезной информации и создание взаимосвязи с конкретным контрольным шаблоном. Это может быть выполнено одним из следующих способов (допускаются и другие способы):

a) хранение полезной информации в управляемой ПБУ базе данных;

b) хранение полезной информации на смарт-карте;

c) хранение полезной информации в непрозрачной части (ББД) ЗБИ.

Примечание — Хранение в ЗБИ возможно только во внутреннем формате ББД. Биометрические данные, соответствующие форматам данных, определенным в ИСО/МЭК 19794, не поддерживают хранение полезной информации.

С.5.3 Полезная информация предоставляется приложению после успешной верификации или идентификации (после выполнения функции ***BioAPI_Verify или BioAPI_VerifyMatch***). ПБУ может использовать политику предоставления полезной информации только в том случае, если достигнутая фактическая ОЛС ниже некоторого порогового значения зафиксированного в записи ПБУ реестра.

С.5.4 Полезной информацией могут быть любые данные, которые являются полезными для приложения; они необязательно должны быть криптографическими, но даже в криптографическом приложении они должны быть представлены именем или защищены ключом. Другие данные, сохраняемые как полезная информация, могут включать в себя пароли, личные идентификационные номера, информацию о привилегии доступа, идентификаторы учетной записи и другую пользовательскую или секретную информацию, которая должна иметь биометрическую защиту.

С.6 Базы данных ЗБИ

С.6.1 Обычно пользовательские базы данных управляются приложением. Во многих случаях пользовательская база данных уже реализована (например, база данных счетов в банке, пользовательский реестр людей, которые принадлежат сетевому домену или имеющих авторизованный доступ к сетевому серверу), и биометрическое приложение только связывает биометрический шаблон с каждым пользователем в базе данных в дополнение к паролю или взамен пароля. Важно, что приложение поддерживает управление доступом к данной базе данных.

С.6.2 БиоАПИ позволяет ПБУ управлять базой данных ЗБИ для обеспечения:

a) оптимизации выполнения операции идентификации по большим совокупностям образцов (то есть когда идентификация осуществляется непосредственно в базе данных);

b) доступа к ЗБИ, которые могут храниться на независимом устройстве регистрации;

c) доступа через ПБУ к ЗБИ, хранящимся в портативной базе данных, например на смарт-карте.

С.6.3 За обеспечение всех необходимых взаимосвязей между базой (базами) данных ЗБИ ПБУ и пользовательской базой (базами) данных отвечает приложение. Для обеспечения данного условия каждая запись в базе данных ЗБИ ПБУ имеет связанный с ней УУИД. Для обеспечения информационной безопасности записи в базе данных ПБУ не могут быть изменены, а могут быть только созданы и удалены. Новые записи получают новые УУИД. УУИД создаются ПБУ каждый раз, когда создается (операцией CreateTemplate или Enroll) или импортируется (операцией Import) шаблон. ПБУ генерирует УУИД и связывает его с контрольным шаблоном в базе данных ЗБИ, управляемой ПБУ.

С.6.4 Не все ПБУ поддерживают идентификацию и интерфейс базы данных. Если совокупность идентификации является небольшой, то она может быть обработана путем передачи массива ЗБИ через интерфейс.

С.6.5 Базы данных ЗБИ могут быть созданы по имени УУИД, а ПБУ может иметь предпочтительную базу данных. Приложение может дать возможность ПБУ выбрать базу данных ЗБИ, в которой будут проводиться операции, с использованием значения пустого указателя на параметр УУИД базы данных.

С.6.6 Управляемая ПБУ база данных ЗБИ может контролироваться непосредственно или косвенно, то есть через интерфейс ПБФ. Сама база данных ЗБИ включает в себя модуль архива БиоАПИ. Несмотря на то, что ПБУ может управлять только одним модулем БиоАПИ в данной прикрепленной сессии, этот модуль может включать в себя многоуровневые базы данных, которые адресуются своими индивидуальными УУИД базы данных.

С.6.7 Для указания текущей записи используются маркеры, которые приблизительно соответствуют «курсору» для реляционных баз данных; однако использование реляционной технологии не требуется и не предполагается.

С.6.8 Новый маркер создает и возвращает дескриптор, когда открывается база данных ЗБИ (при вызове функции ***BioAPI_DbOpen***) или при восстановлении записи (при вызове функции ***BioAPI_DbGetBIR***). Существующий маркер может быть установлен в определенную позицию путем вызова функции ***BioAPI_DbSetMarker.*** Дескриптор маркера может использоваться приложением для упрощения повторного обращения к следующей записи в базе данных ЗБИ. Таким образом, маркер (но не дескриптор маркера) обновляется при вызове функции ***BioAPI_DbGetNextBIR.*** Маркеры и их дескрипторы освобождаются явно при вызове функции ***BioAPI_DbFreeMarker*** или неявно при закрытии базы данных ЗБИ (при вызове функции ***BioAPI_DbClose***).

С.6.9 ПБУ может управлять более чем одной базой данных ЗБИ. В этом случае база данных ЗБИ, управляемая ПБУ, в которой необходимо произвести операции, указывается приложением по имени (УУИД базы данных) или дескриптору (для открытой базы данных).

С.7 Анализ интерфейса пользователя

С.7.1 Интерфейс пользователя для паролей и личных идентификационных номеров является достаточно простым, но для биометрической технологии он может быть достаточно сложным и может в значительной степени зависеть от применяемой технологии, требуя многочисленные зависимые от реализации взаимодействия с пользователем. Некоторые биометрические технологии предоставляют пользователю потоки данных (например, лицо и голос), а другие требуют от пользователя подтверждения каждого взятого образца (например, лицо, голос, подпись). В процессе регистрации некоторые технологии верифицируют каждый взятый образец на основе предыдущих образцов. Число образцов, взятых для конкретного назначения, может меняться в зависимости от используемой технологии, и интерфейс пользователя для регистрации обычно отличается от интерфейсов для верификации и идентификации.

С.7.2 Большинство ПБУ содержат встроенный интерфейс пользователя, которого может быть достаточно для решения большинства задач. Однако ПИП позволяет приложению управлять видом и поведением данного интерфейса пользователя, предоставляя приложению возможность осуществлять обратные вызовы поставщика услуги для их использования, представления и сбора образцов.

С.7.3 Один из обратных вызовов используется для представления и сбора образцов и указывает приложению на изменение состояния. Все ПБУ, реализующие опцию Application Controlled GUI, должны поддерживать данный обратный вызов, несмотря на то что конечные механизмы могут значительно меняться. Другой обратный вызов ГИП используется для представления пользователю потоковых данных в форме последовательности битовых изображений. Данный обратный вызов является необязательным, и ПБУ указывает в своей записи в реестре, требуется ли он.

С.7.4 ПБУ управляется конечным механизмом интерфейса пользователя и инициирует его обратные вызовы (BioAPI_GUI_STATE_CALLBACK) каждый раз, когда возникает событие изменения состояния. Данные изменения состояния могут возникать при создании образца или при необходимости предоставления пользователю сообщения. После возвращения сообщения приложение может предоставить ПБУ ответ пользователя: отмена, продолжение, действительный образец, недействительный образец и т.д.

С.7.5 Если ПБУ должен представить пользователю потоковые данные образца, он инициирует потоковый обратный вызов (BioAPI_GUI_STREAMING_CALLBACK) параллельно с событиями изменения состояния. Данный обратный вызов требует от приложения выполнения многопоточной обработки.

С.7.6 Если приложение выполняет однотипный обратный вызов несколько раз в одном и том же контексте (дескриптор ПБУ), поведение ПБУ будет зависеть от реализации операции. ПБУ выбирает, продолжить ли операцию после возврата первого обратного вызова или ждать возврата всех обратных вызовов. Ситуация, когда различные обратные вызовы возвращаются с различными кодами ответа, также находится под управлением ПБУ.

С.8 Сопоставление на карте (СНК)

С.8.1 Технология БиоАПИ поддерживает использование смарт-карт. Используют два следующих варианта применения:

a) смарт-карта используется в качестве хранилища для контрольного шаблона, что обеспечивает его безопасное и портативное хранение (данный способ называют хранение на карте);

b) операции биометрического сопоставления выполняются на смарт-карте (данный способ называют сопоставление на карте).

С.8.2 В обоих случаях шаблон сохраняется на карте. При сопоставлении на карте (СНК) операции сопоставления также осуществляются на карте. Во время биометрической верификации обработанная ЗБИ передается

на карту, где она сопоставляется с контрольным шаблоном, хранимым на этой карте. Результат сопоставления возвращается ПБУ или ПБФ или используется непосредственно с картой для доступа к другим данным, записанным на карте, и функциям карты (например, для считывания цифровой подписи).

Примечание — При использовании одного из вариантов смарт-карта может выступать в роли модуля архива БиоАПИ или как модуль алгоритмов сопоставления БиоАПИ.

С.8.3 ПБУ и ПБФ могут использовать возможности применения в смарт-картах технологий БиоАПИ согласно ИСО/МЭК 7816-11 [1].

В соответствии с требованиями настоящего стандарта биометрические данные должны быть сохранены в формате постоянного клиента ЕСФОБД, определенном в ИСО/МЭК 19785-3 [4] для сохранения на карте. Для СНК биометрические данные преобразуются в биометрический информационный шаблон (БИШ), который сохраняется на карте и может быть впоследствии запрошен для возврата информации о структуре и содержании ProcessedBIR и послан карте для сопоставления с сохраненным контрольным шаблоном (например, владелец/тип формата и, возможно, другая информация об алгоритмах и шаблоне).

Примечание — Данная информация может быть вспомогательной при использовании функции ***BioAPI_ProcessWithAuxBIR.***

С.8.4 Существует множество способов взаимодействия биометрических компонентов и компонентов смарт-карты (программные и аппаратные средства) для выполнения операции СНК. Один из потоков процесса высокого уровня для такой операции изображен на рисунке С.3. Метод защиты данных верификации с помощью средств криптографии приведен в ИСО/МЭК 7816-11.

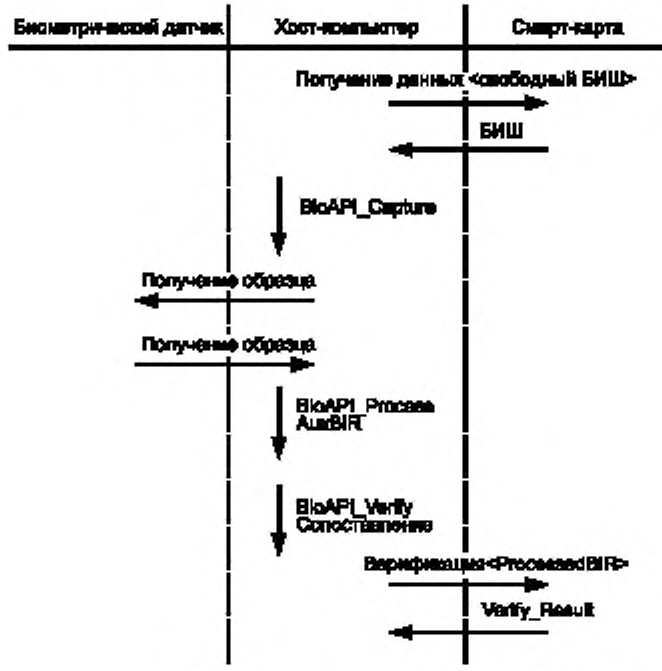


Рисунок С.3 — Пример потока процесса для операции СНК

Примечание 1 — Область, которая расположена выше обмена данными между ПБУ и смарт-картой, описывается как любое взаимодействие с непосредственно управляемым модулем БиоАПИ (смарт-карта выступает в роли модуля архива или модуля алгоритмов сопоставления) или одним или более ПБФ.

Примечание 2 — В данном примере реализации СНК требуется, чтобы ПБУ был совместим со смарт-картой, включал интерфейс взаимодействия со смарт-картой (напрямую с помощью одного или более ПБУ) и поддерживал СНК.

Примечание 3 — Приложение, ПБУ, ПБФ могут взаимодействовать со смарт-картой напрямую или с помощью некого стандартного интерфейса смарт-карты или связующего программного обеспечения. Обычно используется взаимодействие СНК с программным обеспечением, встроенным программным обеспечением или сетевым приложением на смарт-карте.

С.8.5 Операция верификации обычно разделяется между хост-компьютером и смарт-картой. Как правило, получение и обработка биометрических данных реализуются на хост-компьютере, а смарт-карта реализует операции сопоставления и возвращает результат сопоставления.

Примечание 1 — При расширении возможностей смарт-карты на нее могут быть перенесены с хост-компьютера также и другие функции.

Примечание 2 — На смарт-карте может быть установлен личный «статус безопасности», если результат сопоставления положителен.

С.8.6 Некоторые реализации СНК требуют, чтобы часть информации, хранящейся в контрольном шаблоне, была извлечена хост-компьютером для использования в качестве входного параметра операций получения данных или обработки. На рисунке С.3 первый шаг соответствует восстановлению «вспомогательных данных» в формат БИШ.

Примечание 1 — При использовании СНК функция BioAPI_ProcessWithAuxBIR вызывается для обработки полученных данных ЗБИ с использованием вспомогательных данных, предоставленных как входной параметр в соответствии с CapturedBIR.

Примечание 2 — Вспомогательные данные могут включать в себя, например, информацию о регистрации, чтобы предоставить пользователю дополнительную информацию в процессе выполнения функций получения данных или обработки, например расположить палец на рабочей поверхности, чтобы обеспечить пригодность получаемых данных для сопоставлении, прежде чем послать их карте, таким образом снижая общее время верификации.

С.8.7 ПБУ, поддерживающие смарт-карты, могут выполнять операции с ней непосредственно или через ПБФ. Например:

a) хранение ЗБИ на смарт-карте может быть осуществлено через ПБФ архива;

b) верификация СНК может быть осуществлена алгоритмом ПБФ сопоставления.

Приложение D
(справочное)

Примеры последовательности вызовов и типового кода

D.1 Простая последовательность вызовов

БиоАПИ может поддерживать большое разнообразие биометрических технологий и устройств, так же как и методов аутентификации, однако иногда приложению требуется использование только небольшого подмножества вызовов функций для выполнения верификации. В данном разделе указана последовательность вызовов, необходимых для выполнения данного типа верификации.

D.1.1 Инициализация

Сначала необходимо инициализировать инфраструктуру БиоАПИ. Это осуществляется путем вызова функции ***BioAPI_Init.*** Приложение должно определить, с какой версией спецификации БиоАПИ оно совместимо.

```
void function1() {
  BioAPI_VERSION bioVersion;
  BioAPI_RETURN bioReturn;
  #define BioAPI_MAJOR              (2)
  #define BioAPI_MINOR              (0)
  bioVersion.Major = BioAPI_MAJOR;
  bioVersion.Minor = BioAPI_MINOR;
  bioReturn = BioAPI_Init(&bioVersion);
  if(BioAPI_OK != bioReturn)
  {
      if(BioAPIERR_INCOMPATIBLE_VERSION == bioReturn)
      {
        printf("This application is not compatible with the installed version of the BioAPI\n");
          }
          else
          {
              printf("BioAPI Error Code: %d\n", bioReturn);
          }
  }
  }
```

D.1.2 Загрузка и присоединение ПБУ

Если приложение знает УУИД ПБУ, который необходимо использовать, то для подготовки ПБУ к использованию функциями БиоАПИ необходимо только вызвать функции ***BioAPI_BSP_BSPLoad*** и ***BioAPI_BSPAttach.***

Функция ***BioAPI_BSPAttach*** возвращает дескриптор, который будет использоваться при последующих вызовах функций БиоАПИ.

```
typedef struct SensorDetectionContext {
    BioAPI_UNIT_ID unitID;
    BioAPI_BOOL foundSensor;
} SensorDetectionContext;
BioAPI_RETURN BioAPI SensorDetectionEventHandler(
      const BioAPI_UUID *BSPUuid,
      BioAPI_UNIT_ID UnitID,
      void* AppNotifyCallbackCtx,
      const BioAPI_UNIT_SCHEMA *UnitSchema,
      BioAPI_EVENT EventType) {
      SensorDetectionContext *context =
            (SensorDetectionContext *) AppNotifyCallbackCtx;
   if (EventType == BioAPI_NOTIFY_INSERT &&
     UnitSchema != NULL &&
     UnitSchema->UnitCategory == BioAPI_CATEGORY_SENSOR &&
     !context->foundSensor)
     { /* использование первого обнаруженного устройства */
     context->unitID = UnitID;
     context->foundSensor = BioAPI_TRUE;
   }
   return BioAPI_OK;
```

```
}
int function2( ) {
/* ... */
/* Предполагается, что ПБУ выгружается перед возвратом данной функции, в ином случае он оставит неза-
   действованный указатель на контекст от БиоАПИ */
BioAPI_VERSION Version;
BioAPI_RETURN bioReturn;
BioAPI_UUID uuid;
BioAPI_UNIT_LIST_ELEMENT UnitList[1];
BioAPI_HANDLE BSPHandle;
#define BioAPI_MAJOR                (2)
#define BioAPI_MINOR                (0)
SensorDetectionContext context;
context.unitID = 0;
context.foundSensor = BioAPI_FALSE;
bioReturn = BioAPI_BSPLoad(&uuid,
   SensorDetectionEventHandler,
   &context);
   if(BioAPI_OK != bioReturn)
   {
      printf("BioAPI Error Code: %d\n", bioReturn);
      return 0;
   }
   if (context.foundSensor)  {
      Version = (BioAPI_VERSION)((BioAPI_MAJOR << 4) | BioAPI_MINOR);
      UnitList[0].UnitCategory = BioAPI_CATEGORY_SENSOR;
      UnitList[0].UnitId = context.unitID;
      bioReturn = BioAPI_BSPAttach(&uuid,
         Version,
         UnitList,
         1,
         &BSPHandle);
      if(BioAPI_OK != bioReturn)
      {
         printf("BioAPI Error Code: %d\n", bioReturn);
         BioAPI_BSPUnload (&uuid, SensorDetectionEventHandler, &context);
         return 0;
      }
}
 return 0;
}
```

D.1.3 Регистрация субъекта

После подключения ПБУ проводится регистрация биометрических данных субъекта. Это осуществляется путем однократного вызова функции ***BioAPI_Enroll.***

```
int function3() {
  BioAPI_RETURN bioReturn;
  BioAPI_HANDLE BSPHandle;
  BioAPI_BIR_HANDLE EnrolledTemplate;
  bioReturn = BioAPI_Enroll(BSPHandle,
     BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY,
     BioAPI_NO_SUBTYPE_AVAILABLE,
     NULL,
     NULL,
     &EnrolledTemplate,
     NULL,
     -1,
     NULL,
     NULL);
  if(bioReturn != BioAPI_OK)
  {
    printf("BioAPI Error Code: %d\n", bioReturn);
    return 0;
  }
  return 0;
}
```

Как только шаблон будет возвращен, он может быть помещен в хранилище данных для последующего извлечения и верификации (см. D.2).

D.1.4 Выполнение верификации

Для верификации необходимо выполнить несколько вызовов. Если ПБУ поддерживает функцию ***BioAPI_VerifyMatch,*** последовательность вызовов будет более сложной и приложение должно получать данные, создавать шаблон и проводить верификацию поэтапно. В противном случае приложение может просто вызвать функцию ***BioAPI_Verify.***

```
int function4() {
  BioAPI_RETURN bioReturn;
  BioAPI_HANDLE BSPHandle;
  BioAPI_BSP_SCHEMA *CurrSchema;
  BioAPI_BIR_HANDLE EnrolledTemplate, CapturedTemplate, ProcessedBir;
  BioAPI_INPUT_BIR birEnroll, birCapture, InputBirProcessed;
  BioAPI_BIR_HEADER birHeader;
  int MaxFMR, AchievedFMR;
  BioAPI_BOOL bResponse;
  // Определение поддержки функции BioAPI_VerifyMatch ПБУ проводится
  // проверкой operations mask
  if(CurrSchema->Operations & BioAPI_VERIFYMATCH)
{
  if((bioReturn = BioAPI_Capture(BSPHandle,
            BioAPI_PURPOSE_VERIFY,
            BioAPI_NO_SUBTYPE_AVAILABLE,
            NULL,
            &CapturedTemplate,
            -1,
            NULL)) != BioAPI_OK)
  {
      printf("BioAPI Error Code: %d\n", bioReturn);
          return 0;
      }
      if((bioReturn = BioAPI_GetHeaderFromHandle(BSPHandle,
                  CapturedTemplate,
                  &birHeader)) != BioAPI_OK)
                                          {
          printf("BioAPI Error Code: %d\n", bioReturn);
          return 0;
      }
      birEnroll.Form = BioAPI_BIR_HANDLE_INPUT;
      birEnroll.InputBIR.BIRinBSP = &EnrolledTemplate;
      if(birHeader.Type & BioAPI_BIR_DATA_TYPE_INTERMEDIATE)
      {
          birCapture.Form = BioAPI_BIR_HANDLE_INPUT;
          birCapture.InputBIR.BIRinBSP = &CapturedTemplate;
          if((bioReturn = BioAPI_Process(     BSPHandle,
                  &birCapture,
                  NULL,
                  &ProcessedBir)) != BioAPI_OK)
                                              {
              printf("BioAPI Error Code: %d\n", bioReturn);
              return 0;
          }
          MaxFMR = 1;
          InputBirProcessed.Form = BioAPI_BIR_HANDLE_INPUT;
          InputBirProcessed.InputBIR.BIRinBSP = &ProcessedBir;
      }
      else
      {
          MaxFMR = 1;
          InputBirProcessed.Form = BioAPI_BIR_HANDLE_INPUT;
          InputBirProcessed.InputBIR.BIRinBSP = &CapturedTemplate;
      }
```

```
            bioReturn = BioAPI_VerifyMatch(      BSPHandle,
                                MaxFMR,
                                &InputBirProcessed,
                                &birEnroll,
                                NULL,
                                &bResponse,
                                &AchievedFMR,
                                NULL);
        }
        else        // Просто вызываем BioAPI_Verify
        {
            MaxFMR = 1;
            bioReturn = BioAPI_Verify(     BSPHandle,
                                MaxFMR,
                                &birEnroll,
                                BioAPI_NO_SUBTYPE_AVAILABLE,
                                NULL,
                                &bResponse,
                                &AchievedFMR,
                                NULL,
                                -1,
                                NULL);
        }
        if(bioReturn != BioAPI_OK)
        {
            printf("BioAPI Error Code: %d\n", bioReturn);
            return 0;
        }
        if(bResponse == BioAPI_TRUE)
        {
      printf("Match\n");
 }
 else
 {
      printf("No Match\n");
}
return 0;
}
```

D.1.5 Отсоединение и выгрузка ПБУ

Как только приложение закончит использование ПБУ, оно его отсоединяет и выгружает.

```
void function5()  {
BioAPI_RETURN bioReturn;
BioAPI_HANDLE BSPHandle;
BioAPI_UUID uuid;
SensorDetectionContext context;
if(BSPHandle != 0)
{
    bioReturn = BioAPI_BSPDetach(BSPHandle);
    if(BioAPI_OK != bioReturn)
    {
        printf("BioAPI Error Code: %d\n", bioReturn);
        return;
    }
        BSPHandle = 0;
}
bioReturn = BioAPI_BSPUnload (&uuid, SensorDetectionEventHandler, &context);
   /* В реальных приложениях адрес контекста, передаваемый BioAPI_BSPUnload, должен быть аналогичен
      адресу контекста, передаваемому вызову сопоставления BioAPI_BSPLoad call */
if(BioAPI_OK != bioReturn)
{
    printf("BioAPI Error Code: %d\n", bioReturn);
    return;
}
}
```

D.1.6 Закрытие инфраструктуры

Для закрытия инфраструктуры БиоАПИ необходимо вызвать функцию ***BioAPI_Terminate.***

```
void function6() {
BioAPI_Terminate();
}
```

D.2 Преобразование ЗБИ образца

Следующий пример кода преобразует структуру данных C (BioAPI_BIR, определенную в 7.4) в формат постоянного клиента БиоАПИ (определенного в приложении B) и обратно.

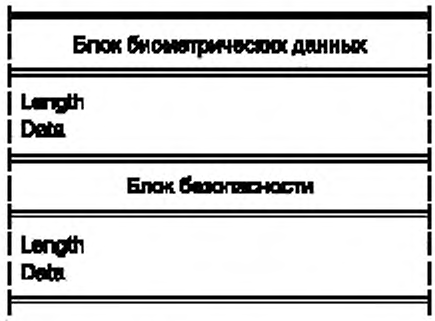
```
/**
 * Sample.c
 */
        #include <stdlib.h>
        #include <string.h>
        #include <stdint.h>
        /**
         * Имя: GetBirHeaderSerializedLength
         * Назначение: возвращает число байтов в заголовке после преобразования в последовательность
                        байтов.
         * Параметры: header [входной] — указатель на заголовок ЗБИ.
         * Возвращаемое значение: число байтов, занимаемых заголовком после преобразования в после-
         *                довательность байтов.
         * Комментарии: возвращаемый размер будет равен 52 байтам, если установлен признак индекса
         *                и индекс включен в заголовок, и 36 байтам, если признак индекса не установлен и
         *                индекс отсутствует.
         *
         *
         *                Возвращаемое значение может отличаться от sizeof (BioAPI_BIR_HEADER) из-за
         *                структурных дополнений, введенных компилятором.
         */
        uint32_t GetBirHeaderSerializedLength(BioAPI_BIR_HEADER_PTR header) {
            uint32_t size =
                sizeof(uint32_t) + /* Длина ЗБИ (4 байта) */
                sizeof(BioAPI_VERSION) + /* Версия заголовка (1 байт) */
                sizeof(BioAPI_BIR_DATA_TYPE) + /* Тип (1 байт) */
                sizeof(uint16_t) + /* Format.FormatOwner (2 байта) */
                sizeof(uint16_t) + /* Format.FormatType (2 байта) */
                sizeof(BioAPI_QUALITY) + /* Качество (1 байт) */
                sizeof(BioAPI_BIR_PURPOSE) + /* Назначение (1 байт) */
                sizeof(BioAPI_BIR_BIOMETRIC_TYPE) + /* FactorsMask (4 байта) */
                sizeof(uint16_t) + /* ProductID.ProductOwner (2 байта) */
                sizeof(uint16_t) + /* ProductID.ProductType (2 байта) */
                4 * sizeof(uint8_t) + /* CreationDTG.Date (4 байта) */
                3 * sizeof(uint8_t) + /* CreationDTG.Time (3 байта) */
                sizeof(BioAPI_BIR_SUBTYPE) + /* Подтип (1 байт) */
                4 * sizeof(uint8_t) + /* ExpirationDate (4 байта) */
                sizeof(uint16_t) + /* SBFormat.SecurityFormatOwner (2 байта) */
                sizeof(uint16_t); /* SBFormat.SecurityFormatType (2 байта) */
            if (header->Type & BioAPI_BIR_INDEX_PRESENT)  {
               size += sizeof(BioAPI_UUID); /* Index (16 bytes) */
            }
            /* общий размер: 36 без индекса; 52 с индексом */
            return size;
        }
        /**
         * Имя: GetBirDataSerializedLength
         * Назначение: возвращает число байтов, требуемых для сохранения длины данных и данных как
         *                ББД, так и SB в виде последовательности байтов.
         * Параметры: data [входной] — указатель на структуру BioAPI_Data.
         *                Возвращаемое значение: число байтов, требуемых для сохранения длины данных и
         *                сами данные в виде последовательности байтов.
```

```
        */
    uint32_t GetBirDataSerializedLength(BioAPI_DATA_PTR data) {
if (NULL == data)
        {
          return sizeof(uint32_t); /* Данные->Длина */
        }
else {
          return sizeof(data->Length) + data->Length;
        }
}
        /**
         * Имя: GetBirSerializedLength
         * Назначение: возвращает число байтов, требуемых для всей информации, хранимой в ЗБИ, в виде
         *                последовательности байтов.
         * Параметры: bir [входной] — указатель на ЗБИ БиоАПИ.
         * Возвращаемое значение: число байтов, требуемых для сохранения всей информации, хранимой в
         *                ЗБИ, в виде последовательности байтов.
         */
        uint32_t GetBirSerializedLength(BioAPI_BIR *bir) {
          uint32_t headerSize = GetBirHeaderSerializedLength(&bir->Header);
          uint32_t bdbSize = GetBirDataSerializedLength(&bir->BiometricData);
          uint32_t sbSize = GetBirDataSerializedLength(&bir->SecurityBlock);
          return headerSize + bdbSize + sbSize;
        }
        /**
         * Имя: SerializeIntToBuffer
         * Назначение: записывает 32-битовое значение в формате сетевого порядка байтов (Big Endian) в
         *                заданный буфер
         * Параметры: buffer [входной] — байтовый буфер, получающий значение
         *                data [входной] — 32-битовое значение, которое должно быть записано в буфер.
         * Возвращаемое значение: местоположение буфера непосредственно за записанным значением.
         *  Комментарии: данная функция автоматически конвертирует формат предоставляемого значения
         *                в сетевой порядок байтов. Значение  не должно быть преобразовано  в сетевой  поря-
         *                док  байтов перед вызовом данной функции.
         */
        uint8_t *SerializeIntToBuffer(uint8_t *buffer, uint32_t data) {
            *buffer++ = (data >> 24) & 0xff;
            *buffer++ = (data >> 16) & 0xff;
            *buffer++ = (data >> 8) & 0xff;
            *buffer++ = (data ) & 0xff;
            return buffer;
        }
        /**
         * Имя: ExtractIntFromBuffer
         * Назначение: читает 32-битовое значение  в  формате  сетевого  порядка  байтов  (Big Endian)  из
         *                заданного буфера.
         * Параметры: buffer [входной] — байтовый буфер, содержащий значения;
         *                data [выходной] — 32-битовое значение, которое должно быть извлечено из буфера.
         * Возвращаемое значение: местоположение буфера непосредственно за прочитанным значением.
         * Комментарии: данная функция автоматически конвертирует формат предоставляемого значения
                        в основной порядок байтов из сетевого.
         */
        uint8_t *ExtractIntFromBuffer(uint8_t *buffer, uint32_t *data) {
          /* преобразование данных в системный порядок байтов */
          *data = (buffer[0] << 24) | (buffer[1] << 16) | (buffer[2] << 8) | buffer[3];
          /* перемещение буфера за прочитанную информацию */
          buffer += sizeof(networkData);
          return buffer;
```

```
}
/**
 * Имя: SerializeWordToBuffer
 * Назначение: записывает 16-битовое значение в формате сетевого порядка байт (Big Endian) в
 *             заданный буфер.
 * Параметры: buffer [входной] — байтовый буфер, принимающий значение;
 *             data [входной] — 16-битовое значение, предназначенное для записи в буфер.
 * Возвращаемое значение: местоположение буфера непосредственно за записанным значением.
 * Комментарии: Данная функция автоматически конвертирует предоставляемое значение в формат
 *             сетевого порядка байтов. Значение не должно быть преобразовано в формат сете-
 *             вого порядка перед вызовом
 *             данной функции.
 */
uint8_t *SerializeWordToBuffer(uint8_t *buffer, uint16_t data) {
  *buffer++ = (data >> 8) & 0xff;
  *buffer++ = (data ) & 0xff;
  return buffer;
}
/**
 * Имя: ExtractWordFromBuffer
 * Назначение: читает 16-битовое значение в формате сетевого порядка байтов (Big Endian) из
 *             заданного буфера.
 * Параметры: buffer [входной] — байтовый буфер, содержащий значение;
 *             data [выходной] — 16-битовое значение, которое должно быть извлечено из буфера.
 * Возвращаемое значение: местоположение буфера непосредственно за прочитанным значением.
 * Комментарии: данная функция автоматически конвертирует формат предоставленного значения
 * в основной порядок байтов из сетевого.
 */
uint8_t *ExtractWordFromBuffer(uint8_t *buffer, uint16_t *data) {
    /* преобразует данные в системный порядок байтов */
    *data = (buffer[0] << 8) | buffer[1];
    return buffer;
}
/**
 * Имя: SerializeByteToBuffer
 * Назначение: записывает значения байтов в заданный буфер.
 * Параметры: buffer [входной] — байтовый буфер, получающий значение;
 *             data [входной] — 8-битовое значение, предназначенное для записи в буфер.
 * Возвращаемое значение: местоположение буфера непосредственно за записанным значением.
 */
uint8_t *SerializeByteToBuffer(uint8_t *buffer, uint8_t data) {
    /* копирует данные в буфер */
    * buffer = data,
    /* перемещает буфер в положение непосредственно за записанной информацией */
    buffer++;
    return buffer;
}
/**
 * Имя: ExtractByteFromBuffer
 * Назначение: читает один байт из заданного буфера
 * Параметры: buffer [входной] — байтовый буфер, содержащий значение;
 *             data [выходной] — 8-битовое значение, которое должно быть извлечено из буфера.
 * Возвращаемое значение: местоположение буфера непосредственно за прочитанным значением.
 */
uint8_t *ExtractByteFromBuffer(uint8_t *buffer, uint8_t *data) {
    /* копирует данные из буфера */
    *data = *buffer;
    /* перемещает буфер в положение непосредственно за прочитанной информацией */
    buffer++;
    return buffer;
```

```
}
/**
 * Имя: SerializeByteArrayToBuffer
 * Назначение: записывает массив байтов в заданный буфер.
 * Параметры: buffer [входной] — байтовый буфер, получающий значение;
 *            data [входной] — данные, которые будут записаны в буфер;
 *            length [входной] — длина данных, которые будут записаны в буфер.
 * Возвращаемое значение: месторасположение буфера непосредственно за записанным значением.
 */
uint8_t *SerializeByteArrayToBuffer(uint8_t *buffer, uint8_t *data, uint32_t length) {
    /* копирует данные в буфер */
    memcpy(buffer, data, length);
    /* перемещает буфер в положение непосредственно за записанной информацией */
    buffer += length;
    return buffer;
}
/**
 * Имя: ExtractByteArrayFromBuffer
 * Назначение: читает массив байтов из заданного буфера.
 * Параметры: buffer [входной] — байтовый буфер, содержащий значение;
 *            data [выходной] — данные, которые будут прочитаны из буфера;
 *            length [входной] — длина данных, которые будут прочитаны из буфера.
 * Возвращаемое значение: местоположение буфера непосредственно за прочитанной информацией.
 */
uint8_t *ExtractByteArrayFromBuffer(uint8_t *buffer, uint8_t *data, uint32_t length) {
    /* копирует данные из буфера */
    memcpy(data, buffer, length);
    /* перемещает буфер в положение непосредственно за прочитанной информацией */
    buffer += length;
    return buffer;
}
/**
 * Имя: SerializeBirToBuffer
 * Назначение: записывает всю информацию, содержащуюся в ЗБИ БиоАПИ, в заданный буфер.
 * Параметры: buffer [входной] — байтовый буфер, получающий значение;
 *            bir [входной] — указатель на данные ЗБИ.
 * Возвращаемое значение: местоположение буфера непосредственно за записанным значением.
 * Комментарии: сериализованный формат содержит все целочисленные значения в формате сетевого порядка байтов
 *            (Big Endian).
 *
 *  ╠═══════════════════════════════╣
 *  ┃          Заголовок            ┃
 *  ╠═══════════════════════════════╣
 *  | Stored BIR Length in bytes    | (4 байта)
 *  | Version                       | (1 байт)
 *  | Type                          | (1 байт)
 *  | Format Owner                  | (2 байта)
 *  | Format Type                   | (2 байта)
 *  | Quality                       | (1 байт)
 *  | Purpose                       | (1 байт)
 *  | Factors Mask                  | (4 байта)
 *  | Product Owner                 | (2 байта)
 *  | Product Type                  | (2 байта)
 *  | CreationDTG Year              | (2 байта)
 *  | CreationDTG Month             | (1 байт)
 *  | CreationDTG Day               | (1 байт)
 *  | CreationDTG Hour              | (1 байт)
 *  | CreationDTG Minute            | (1 байт)
 *  | CreationDTG Second            | (1 байт)
```

```
 * | SubType                          | (1 байт)
 * | ExpirationDate Year              | (2 байта)
 * | ExpirationDate Month             | (1 байт)
 * | ExpirationDate Day               | (1 байт)
 * | SB Format Owner                  | (2 байта)
 * | SB Format Type                   | (2 байта)
 * | Index                            | (16 байтов, необязательно, контролируется
 * |                                  | признаком индекса в поле Type)
```




```
 */
uint8_t *SerializeBirToBuffer(uint8_t *buffer, BioAPI_BIR *bir) {
    uint32_t birSize = GetBirSerializedLength(bir);
    uint32_t storedBIRLength = birSize - sizeof(uint32_t);
      /* Длина поля не включает в себя определение */
      /* сохраненной длины ЗБИ */
    /*
    * Сериализация заголовка в буфер по принципу поле за один  раз. Преобразует 16- и 32-битовые
    *                целочисленные значения в сетевой порядок байтов.
    */
  buffer = SerializeIntToBuffer(buffer, storedBIRLength);
  buffer = SerializeByteToBuffer(buffer, bir->Header.HeaderVersion);
  buffer = SerializeByteToBuffer(buffer, bir->Header.Type);
  buffer = SerializeWordToBuffer(buffer, bir->Header.Format.FormatOwner);
  buffer = SerializeWordToBuffer(buffer, bir->Header.Format.FormatType);
  buffer = SerializeByteToBuffer(buffer, bir->Header.Quality);
  buffer = SerializeByteToBuffer(buffer, bir->Header.Purpose);
  buffer = SerializeIntToBuffer(buffer, bir->Header.FactorsMask);
  buffer = SerializeWordToBuffer(buffer, bir->Header.ProductID.ProductOwner);
  buffer = SerializeWordToBuffer(buffer, bir->Header.ProductID.ProductType);
  buffer = SerializeWordToBuffer(buffer, bir->Header.CreationDTG.Date.Year);
  buffer = SerializeByteToBuffer(buffer, bir->Header.CreationDTG.Date.Month);
  buffer = SerializeByteToBuffer(buffer, bir->Header.CreationDTG.Date.Day);
  buffer = SerializeByteToBuffer(buffer, bir->Header.CreationDTG.Time.Hour);
  buffer = SerializeByteToBuffer(buffer, bir->Header.CreationDTG.Time.Minute);
  buffer = SerializeByteToBuffer(buffer, bir->Header.CreationDTG.Time.Second);
  buffer = SerializeByteToBuffer(buffer, bir->Header.Subtype);
  buffer = SerializeWordToBuffer(buffer, bir->Header.ExpirationDate.Year);
  buffer = SerializeByteToBuffer(buffer, bir->Header.ExpirationDate.Month);
  buffer = SerializeByteToBuffer(buffer, bir->Header.ExpirationDate.Day);
  buffer = SerializeWordToBuffer(buffer, bir->Header.SBFormat.SecurityFormatOwner);
  buffer = SerializeWordToBuffer(buffer, bir->Header.SBFormat.SecurityFormatType);
    /* Дополнительно включает индекс */
    if (bir->Header.Type & BioAPI_BIR_INDEX_PRESENT) {
      buffer = SerializeByteArrayToBuffer(buffer,
        bir->Header.Index, sizeof(bir->Header.Index));
    }
    /* Сериализует ББД в буфер */
    buffer = SerializeIntToBuffer(buffer, bir->BiometricData.Length);
```

```
    if (bir->BiometricData.Length > 0)
        buffer = SerializeByteArrayToBuffer(buffer,
            (uint8_t *)bir->BiometricData.Data, bir->BiometricData.Length);
    /* Сериализует SB в буфер */
    buffer = SerializeIntToBuffer(buffer, bir->SecurityBlock.Length);
    if (bir->SecurityBlock.Length > 0)
        buffer = SerializeByteArrayToBuffer(buffer,
            (uint8_t *)bir-> SecurityBlock.Data, bir->SecurityBlock.Length);
    return buffer;
}
/**
 * Имя: ExtractBirFromBuffer
 * Назначение: извлекает всю информацию о ЗБИ БиоАПИ из заданного буфера, ранее записанную
 *             функцией SerializeBIRToBuffer.
 * Параметры: buffer [входной] — байтовый буфер, содержащий значение;
 *            BIR [выходной] — указатель на полученные данные ЗБИ.
 * Возвращаемое значение: местоположение буфера непосредственно за извлеченной ЗБИ.
 * Комментарии: см. описание сериализованного формата в SerializeBIRToBuffer. Возвращаемая
 *             ЗБИ должна быть освобождена с помощью FreeExtractedBIR. При ошибке выделения
 *             памяти возвращаемая ЗБИ имеет пустой указатель и возвращается первоначальное
 *             местоположение буфера, указанное во входном параметре.
 */
uint8_t *ExtractBirFromBuffer(uint8_t *buffer, BioAPI_BIR **bir) {
    uint32_t storedBIRLength;
    /* первоначальное местоположение буфера для обработки ошибок памяти */
    uint8_t *originalBuffer = buffer;
    /* содержит реконструированную ЗБИ */
    BioAPI_BIR *resultBir = (BioAPI_BIR *) malloc(sizeof(BioAPI_BIR));
    if (NULL == resultBir) {
        *bir = NULL;
        return originalBuffer;
    }
    /*
    * Извлекает заголовок из буфера. Автоматически преобразует 16- и 32-битовые целочисленные
    *             значения из формата сетевого порядка байтов.
    */
    buffer = ExtractIntFromBuffer(buffer, &storedBIRLength);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.HeaderVersion);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.Type);
    buffer = ExtractWordFromBuffer(buffer, &resultBir->Header.Format.FormatOwner);
    buffer = ExtractWordFromBuffer(buffer, &resultBir->Header.Format.FormatType);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.Quality);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.Purpose);
    buffer = ExtractIntFromBuffer(buffer, &resultBir->Header.FactorsMask);
    buffer = ExtractWordFromBuffer(buffer, &resultBir->Header.ProductID.ProductOwner);
    buffer = ExtractWordFromBuffer(buffer, &resultBir->Header.ProductID.ProductType);
    buffer = ExtractWordFromBuffer(buffer, &resultBir->Header.CreationDTG.Date.Year);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.CreationDTG.Date.Month);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.CreationDTG.Date.Day);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.CreationDTG.Time.Hour);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.CreationDTG.Time.Minute);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.CreationDTG.Time.Second);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.Subtype);
    buffer = ExtractWordFromBuffer(buffer, &resultBir->Header.ExpirationDate.Year);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.ExpirationDate.Month);
    buffer = ExtractByteFromBuffer(buffer, &resultBir->Header.ExpirationDate.Day);
    buffer = ExtractWordFromBuffer(buffer,
    &resultBir->Header.SBFormat.SecurityFormatOwner);
    buffer = ExtractWordFromBuffer(buffer,
    &resultBir->Header.SBFormat.SecurityFormatType);
```

```
    /*
       Дополнительно извлекает индекс.
    */
    if (resultBir->Header.Type & BioAPI_BIR_INDEX_PRESENT) {
       buffer = ExtractByteArrayFromBuffer(buffer,
          resultBir->Header.Index, sizeof(resultBir->Header.Index));
    }
    else {
        memset(resultBir->Header.Index, 0, sizeof(resultBir->Header.Index));
    }
    /* Извлекает длину ББД */
    buffer = ExtractIntFromBuffer(buffer, &resultBir->BiometricData.Length);
    /* Распределяет данные ББД */
    resultBir->BiometricData.Data = malloc(resultBir-> BiometricData.Length);
    if (NULL == resultBir->BiometricData.Data) {
        free(resultBir);
        *bir = NULL;
        return originalBuffer;
    }
    /* Извлекает данные ББД */
    buffer = ExtractByteArrayFromBuffer(buffer,
        (uint8_t *)resultBir->BiometricData.Data,
        resultBir->BiometricData.Length);
    /* Извлекает длину SB */
    buffer = ExtractIntFromBuffer(buffer, &resultBir->SecurityBlock.Length);
    if (resultBir->SecurityBlock.Length > 0) {
        /* Распределяет данные SB */
        resultBir->SecurityBlock.Data =
            malloc(resultBir->SecurityBlock.Length);
        if (NULL == resultBir->SecurityBlock.Data) {
        free(resultBir->BiometricData.Data);
            free(resultBir);
            *bir = NULL;
            return originalBuffer;
        }
        /* Извлекает данные SB */
        buffer = ExtractByteArrayFromBuffer(buffer,
            (uint8_t *) resultBir->SecurityBlock.Data,
            resultBir->SecurityBlock.Length);
    }
    else {
        /* Пустой блок безопасности */
        resultBir->SecurityBlock.Data = NULL;
    }
    /* Возвращает извлеченную ЗБИ */
    *bir = resultBir;
    return buffer;
}
/**
 * Имя: FreeExtractedBir
 * Назначение: освобождает память, выделенную функцией ExtractBirFromBuffer
 * Параметры: BIR [входной] — указатель на данные ЗБИ, которые должны быть освобождены.
 * Возвращаемое значение: нет.
 */
void FreeExtractedBir(BioAPI_BIR *bir) {
    if (NULL != bir) {
        if (NULL != bir->SecurityBlock.Data) {
            free(bir->SecurityBlock.Data);
            bir->SecurityBlock.Data = NULL;
        }
        if (NULL != bir->BiometricData.Data) {
            free(bir->BiometricData.Data);
            bir->BiometricData.Data = NULL;
        }
```

```
            free(bir);
        }
    }
    /**
     * Имя: StoreAndRetrieveBir
     * Назначение: показывает функции хранения ЗБИ.
     * Параметры: ЗБИ [входной] — указатель на данные ЗБИ, над которыми совершаются операции.
     * Возвращаемое значение: нет.*/
    void StoreAndRetrieveBir(BioAPI_BIR *bir) {
        /* Буфер для хранения ЗБИ */
        uint8_t *buffer;
        /* Длина буфера хранения */
        uint32_t bufferLength;
        /* Извлеченная ЗБИ */
        BioAPI_BIR *retrievedBIR;
        /* Рассчитывает длину буфера, необходимую для сохранения ЗБИ */
        bufferLength = GetBirSerializedLength(bir);
        /* выделяет буфера */
        buffer = (uint8_t *) malloc(bufferLength);
        if (NULL == buffer) {
            /* abort */
            return;
        }
        /* записывает ЗБИ в буфер */
        SerializeBirToBuffer(buffer, bir);
        /* ... PLACEHOLDER: записывает буфер в хранилище или передает в сеть... */
        /* ... PLACEHOLDER: извлекает буфер ЗБИ из хранилища или из сети... */
        /* извлекает ЗБИ из буфера */
        ExtractBirFromBuffer(buffer, &retrievedBIR);
        if (NULL != retrievedBIR) {
            /* ... PLACEHOLDER: использует извлеченную ЗБИ ... */
            /* освобождает извлеченную ЗБИ */
        FreeExtractedBir(retrievedBIR);
        }
        free(buffer);
        }
```

Приложение Е
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам

Сведения о соответствии ссылочных международных стандартов национальным стандартам приведены в таблице Е.1.

Таблица Е.1 — Сведения о соответствии ссылочных международных стандартов национальным стандартам

Обозначение ссылочного международного стандарта	Обозначение и наименование соответствующего национального стандарта
ИСО/МЭК 9899:1999	*
ИСО/МЭК 24709	*
ИСО/МЭК 19785-1	ГОСТ Р ИСО/МЭК 19785-1 — 2007 «Автоматическая идентификация. Идентификация биометрическая. Единая структура форматов обмена биометрическими данными. Часть 1. Спецификация элементов данных»
ИСО/МЭК 19785-2	ГОСТ Р ИСО/МЭК 19785-2 — 2007 «Автоматическая идентификация. Идентификация биометрическая. Единая структура форматов обмена биометрическими данными. Часть 2. Спецификация элементов данных»
ИСО/МЭК 19092	*
ИСО 8601	*
ИСО/МЭК 9834-8	*
ИСО/МЭК 10646	*
ИСО/МЭК 19794	

* Соответствующий национальный стандарт отсутствует. Оригинал международного стандарта ИСО/МЭК находится в Федеральном информационном фонде технических регламентов и стандартов.

Библиография

[1] ISO/IEC 7816-11:2004	Identification cards — Integrated circuit cards — Part 11: Personal verification through biometric methods
[2] ISO 8601	Data elements and interchange formats — Information interchange — Representation of dates and times
[3] ISO 19092	Financial services — Biometrics*
[4] ISO/IEC 19785-3	Information technology — Common biometric exchange formats framework — Part 3: Patron format specifications*
[5] ISO/IEC 19794	Information technology — Biometric data interchange formats
[6] ISO/IEC 24708	Information technology — BioAPI Interworking Protocol*
[7] ISO/IEC 24709	Information technology — Conformance testing for BioAPI*

* *Будет опубликован.*

УДК 004.93'1:006.354 ОКС 35.040 П85

Ключевые слова: автоматическая идентификация, идентификация биометрическая, спецификация БиоАПИ, программный интерфейс приложения, ПБУ

Редактор *Т. А. Леонова*
Технический редактор *В. Н. Прусакова*
Корректор *В. И. Варенцова*
Компьютерная верстка *А. П. Финогеновой*

Сдано в набор 01.09.2008. Подписано в печать 16.12.2008. Формат 60×84 1/8. Бумага офсетная. Гарнитура Ариал. Печать офсетная. Усл. печ. л. 14,42. Уч.-изд. л. 15,50. Тираж 141 экз. Зак. 1963.

ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.
www.gostinfo. ru info@gostinfo. ru
Набрано и отпечатано в Калужской типографии стандартов, 248021 Калуга, ул. Московская, 256

Изменение № 1 ГОСТ Р ИСО/МЭК 19784-1—2007 Автоматическая идентификация. Идентификация биометрическая. Биометрический программный интерфейс. Часть 1. Спецификация биометрического программного интерфейса

Утверждено и введено в действие Приказом Федерального агентства по техническому регулированию и метрологии от 23.11.2010 № 491-ст

Дата введения 2012—07—01

Введение. Первый абзац изложить в новой редакции:

«Настоящий стандарт определяет высокоуровневую обобщенную модель биометрического распознавания, пригодную для использования в любой биометрической технологии. Настоящий стандарт не обеспечивает поддержку мультимодальной биометрии в явном виде»;

дополнить абзацами (после третьего):

«Настоящий стандарт определяет поведение инфраструктуры БиоАПИ, когда приложения и ПБУ находятся в одной системе. Другие взаимодействующие стандарты (см. 4.29) определяют разновидности поведения, которые позволяют ПБУ и графическому интерфейсу пользователя работать удаленно от системы, содержащей приложение.

Примечание — ИСО/МЭК 24708 [6], определяющий протокол межсетевого обмена биометрического программного интерфейса (ПМО БиоАПИ), является примером стандарта межсетевого обмена».

Раздел 3 дополнить ссылкой:

«IETF RFC 3987 Интернационализированные идентификаторы ресурсов (ИИР)».

Раздел 4 дополнить подразделами — 4.29 — 4.31:

«4.29 **стандарты межсетевого обмена** (interworking standards): Стандарты, которые преобразовывают действия инфраструктуры БиоАПИ (и требования для соответствия БиоАПИ) для поддержки использования коммуникационных каналов связи с целью предоставления возможности приложению взаимодействовать с удаленным ПБУ с использованием стандартизированного протокола.

4.30 **тестировать-верифицировать/тест-верификация** (test-verify/test-verification): Процесс сравнения «один к одному» тестового образца с биометрическим шаблоном при регистрации для определения соответствия тестового образца биометрическому шаблону.

4.31 **тип регистрации** (enroll type): Значение, которое указывает шаблон подопераций, выполняемых ПБУ в процессе операции регистрации».

Раздел 5 после сокращения «ИД» дополнить сокращением: «ИИР — интернационализированный идентификатор ресурса (см. RFC 3987);

ПМО БиоАПИ — протокол межсетевого обмена биометрического программного интерфейса».

Пункт 6.1.8 дополнить перечислением — е):

«e) предоставление приложению графической информации об активной операции регистрации, верификации или идентификации».

Пункт 6.3.6. Перечисления a), b), c) после обозначения ***«BioAPI_Init»*** дополнить словами: «или ***BioAPI_InitEndpoint*** (только в БиоАПИ 2.1)».

Подраздел 7.8. Заголовок изложить в новой редакции:

«7.8 Тип BioAPI_BIR_BIOMETRIC_TYPE (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии БиоАПИ 2.0».

Подраздел 7.14. Заголовок изложить в новой редакции:

«7.14 Тип BioAPI_BIR_SUBTYPE (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии БиоАПИ 2.0».

Подраздел 7.16. Заголовок изложить в новой редакции:

«7.16 Тип BioAPI_BSP_SCHEMA (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии БиоАПИ 2.0».

Подраздел 7.27. Примечание. Заменить слова: «Иногда невозможно определить» на «Когда используется БиоАПИ версии 2.0, иногда невозможно определить».

Пункт 7.28.2 изложить в новой редакции:

«7.28.2 Определения

BSPUuid(входной) — УУИД ПБУ, инициирующего событие;

UnitID (входной) — ИД модуля БиоАПИ, связанного с событием;

AppNotifyCallbackCtx (входной) — общий указатель на контекстную информацию, которая предоставляется при вызове функции ***BioAPI_BSPLoad,*** устанавливающей обработчик событий;

UnitSchema (входной) — указатель на схему модуля БиоАПИ, связанного с данным событием;

EventType (входной) — тип произошедшего события BioAPI_ EVENT».

Подраздел 7.31. Заголовок изложить в новой редакции:

«7.31 Тип BioAPI_GUI_BITMAP (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии БиоАПИ 2.0».

Подраздел 7.32. Заголовок изложить в новой редакции:

«7.32 Тип BioAPI_GUI_MESSAGE (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии стандарта БиоАПИ 2.0».

Подраздел 7.33. Заголовок изложить в новой редакции:

«7.33 Тип BioAPI_GUI_PROGRESS (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии БиоАПИ 2.0».

Подраздел 7.34. Заголовок изложить в новой редакции:

«7.34 Тип BioAPI_GUI_RESPONSE (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии БиоАПИ 2.0».

Подраздел 7.35. Заголовок изложить в новой редакции:

«7.35 Тип BioAPI_GUI_STATE (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии БиоАПИ 2.0».

Подраздел 7.36. Заголовок изложить в новой редакции:

«7.36 Тип BioAPI_GUI_STATE_CALLBACK (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании версии БиоАПИ 2.0».

Подраздел 7.37. Заголовок изложить в новой редакции:

«7.37 Тип BioAPI_GUI_STREAMING_CALLBACK (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подраздел применяется только при использовании номера версии БиоАПИ 2.0».

Подраздел 7.38 дополнить абзацами:

«При использовании БиоАПИ версии 2.1 нулевое значение не должно использоваться в качестве действительного значения дескриптора присоединенной сессии ПБУ. Все остальные 32-битовые беззнаковые целочисленные значения разрешены.

#define BioAPI_INVALID_BSP_HANDLE (0)

Данное определение применимо только при использовании БиоАПИ версии 2.1».

Подраздел 7.46 после абзаца «#define BioAPI_SETGUICALLBACKS (0x00000002)»

дополнить абзацем:

«#define BioAPI_SUBSCRIBETOGUIEVENTS (0x00000002)»;

после абзаца «#defineBioAPI_CONTROLUNIT (0X00400000)»

дополнить абзацем:

«#define BioAPI_TRANSFORM (0x00800000)»;

дополнить абзацами (перед примечанием):

«Определение BioAPI_SETGUICALLBACKS применяется только при использовании версии БиоАПИ 2.0. Определение BioAPI_SUBSCRIBETOGUIEVENTS применяется только при использовании версии БиоАПИ 2.1.

Данное определение BioAPI_TRANSFORM применимо только при использовании БиоАПИ версии 2.1».

Подраздел 7.47. Заменить слова и дополнить абзацами: «Если установлено, то ПБУ предоставляет потоковые данные ГИП» на

«Если задано, то ПБУ предоставляет потоковые данные ГИП. Данное определение применяется только при использовании версии БиоАПИ 2.0.

#define BioAPI_GUI_PROGRESS_EVENTS (0x00000020)

Если задано, то ПБУ предоставляет потоковые данные ГИП. Данное определение применяется только при использовании версии БиоАПИ 2.1.

#define BioAPI_IDENTIFYINDICATOR (0x00200000)

Если задано, то ПБУ поддерживает индикацию выполнения в процессе идентификации. Данное определение применяется только при использовании версии БиоАПИ 2.1».

Подраздел 7.57 (7.57.1, 7.57.2*) изложить в новой редакции (сноску * исключить):

«7.57 Тип BioAPI_VERSION

Данный тип используется для представления версии БиоАПИ или определения ИПФ (Интерфейс Поставщика Функции), для которой реализованы компоненты или данные. Данный тип используется в функции BioAPI_Init, в заголовке ЗБИ и в схемах реестра компонентов.

Следующие два значения определены в настоящем стандарте:

(1) 32 десятеричное (20 шестнадцатеричное), соответствующее значению номера редакции 2 и значению номера поправки 0 и представляющее БиоАПИ 2.0; и

(2) 33 десятеричное (21 шестнадцатеричное), соответствующее значению номера редакции 2 и значению номера поправки 1 и представляющее БиоАПИ 2.1.

typedef uint8_tBioAPI_VERSION;

Примечание 1 – Данный тип не используется для версий продукта, которые обычно представлены строками.

Примечание 2 – Версия БиоАПИ, используемая в рамках заголовка ЗБИ, соответствует «CBEFF_patron_header_version» в ИСО/МЭК 19785-1

Версия БиоАПИ является соединением значения номера редакции и значения номера поправки так, что первое шестнадцатеричное число представляет номер редакции, а второе шестнадцатеричное число — номер поправки:

BioAPI_VERSION 0xnm,
где n — номер редакции,
m — номер поправки».

Раздел 7 дополнить подразделами и пунктами — 7.58; 7.59, 7.59.1—7.59.10; 7.60, 7.60.1—7.60.4; 7.61, 7.61.1—7.61.3; 7.62; 7.63; 7.64; 7.65; 7.66; 7.67; 7.68; 7.69; 7.70; 7.71, 7.71.1—7.71.3; 7.72:

«7.58 Тип BioAPI_BIR_BIOMETRIC_TYPE (БиоАПИ 2.1)

Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

Маска, которая описывает набор биометрических типов (факторов), содержащихся в ЗБИ БиоАПИ или поддерживаемых ПБУ.

```
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)
```

Примечание 1 — BioAPI_TYPE_MULTIPLE_BIOMETRIC_TYPE используется для обозначения того, что биометрические образцы, содержащиеся в ББД (ЗБИ), включают в себя образцы, полученные от биометрических сканеров разных типов (например, данные отпечатков пальцев и изображения лица). Расположение индивидуальных образцов в ББД определяет владелец формата и идентифицируется значением типа формата.

Примечание 2 — Значение NO VALUE AVAILABLE указывается установкой нулевого значения. Данное значение должно использоваться в том случае, если для ЗБИ, которые первоначально не были созданы ПБУ БиоАПИ,

а были преобразованы в ЗБИ БиоАПИ, информация о биометрическом типе недоступна в записи первоначального источника. Преобразованные ЗБИ, чьи биометрические типы не соответствуют ни одному из определенных типов, должны использовать значение OTHER.

Примечание 3 – Биометрический тип ЗБИ БиоАПИ соответствует «CBEFF_BDB_biometric_type» по ИСО/МЭК 19785-1.

Примечание 4 – Несмотря на то, что password не является биометрической характеристикой, BioAPI_TYPE_PASSWORD включен как действительный BioAPI_BIR_BIOMETRIC_TYPE для поддержки использования password: a) для разработки и тестирования, b) как дополнительный фактор аутентификации в рамках приложения БиоАПИ.

Примечание 5 – Наименования нескольких биометрических типов в версии БиоАПИ 2.1 отличается от имен в версии БиоАПИ 2.0, но без изменений в семантике. Также в версию БиоАПИ 2.1 включено несколько новых биометрических типов. Наконец, «температурные» типы присутствуют в версии БиоАПИ 2.0, но отсутствуют в версии БиоАПИ 2.1. Данные различия внесены для соответствия с ИСО/МЭК 19785-1.

Примечание 6 – Наименования следующих значений не совпадают в версии БиоАПИ 2.0 и БиоАПИ 2.1, но их семантика не изменена. Наименования в версии БиоАПИ 2.1 соответствуют приведенным в ИСО/МЭК 19785-1 (ЕСФОБД).

Наименование в версии БиоАПИ 2.0	Наименование в версии БиоАПИ 2.1	Кодировка
MULTIPLE	MULTIPLE_BIOMETRIC_TYPES	0x00000001
FACIAL_FEATURES (ЧЕРТЫ ЛИЦА)	FACE (ЛИЦО)	0x00000002
FINGERPRINT (ОТПЕЧАТОК ПАЛЬЦА)	FINGER (ПАЛЕЦ)	0x00000008
SIGNATURE_DYNAMICS (ДИНАМИКА ПОДПИСИ)	SIGNATURE_SIGN (ЗНАК ПОДПИСИ)	0x00000080
KEYSTROKE_DYNAMICS (ДИНАМИКА РАБОТЫ С КЛАВИАТУРЫ)	KEYSTROKE (РАБОТА С КЛАВИАТУРОЙ)	0x00000100

Примечание 7 – Следующие значения не присутствуют в версии БиоАПИ 2.0, но присутствуют в версии БиоАПИ 2.1 для совместимости с ИСО/МЭК 19785-1 (ЕСФОБД).

Добавленное значение	Кодировка
NO_BIOTYPE_AVAILABLE	0x00000000
VEIN (ВЕНА)	0x00002000
DNA (ДНК)	0x00004000
EAR (УХО)	0x00008000
FOOT (СТУПНЯ)	0x00010000
SCENT (ЗАПАХ)	0x00020000

7.59 Тип BioAPI_BIR_SUBTYPE (БиоАПИ 2.1)

7.59.1 Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

7.59.2 BioAPI_BIR_SUBTYPE идентифицирует подтип в рамках типа ББД (блок биометрических данных) (определенный в BioAPI_BIR_ BIOMETRIC_TYPE). Его значения и содержание определены дескриптором типа ББД.

Примечание — В определении данного типа в версии БиоАПИ 2.1 могут быть установлены несколько битов. Данное определение соответствует ИСО/МЭК 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)
```

7.59.3 Данная структура является битовой маской с битами, определенными в таблице, приведенной ниже. Бит 7 используется для указания интерпретации битов более низкого порядка. Позиция бита ноль (0) или один (1) всегда указывают право и лево соответственно. Если позиция бита 7 не установлена, то данные позиции битов могут применяться к любому биометрическому типу; однако позиции битов 2—6 характерны

для биометрических типов «палец» и «вена». Если позиция бита 7 установлена, то остальные позиции битов используются только для биометрического типа «вена».

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
0 (любой)	Мизинец	Безымянный	Средний	Указательный	Большой	Правый	Левый
1 (только вена)	Зарезервирован	Зарезервирован	Запястье	Тыльная сторона ладони	Ладонь	Правый	Левый

Примечание – Биометрические данные рисунка вен могут быть извлечены из изображений пальцев (Бит 7 задан равным нулю), или от запястья, ладони, или тыльной стороны ладони правой или левой руки (Бит 7 задан равным 1).

7.59.4 Может быть установлено ноль или больше битов. Абстрактное значение NO VALUE AVAILABLE указывается установкой всех битов на ноль.

Примечание – Абстрактное значение NO VALUE AVAILABLE может быть использовано ЗБИ, которые не были созданы ПБУ БиоАПИ, а были преобразованы из другого формата данных. Данное значение также может быть использовано, когда ни одно другое значение не применимо или информация недоступна.

7.59.5 Позиции битов с BioAPI_BIR_SUBTYPE_THUMB по BioAPI_BIR_SUBTYPE_LITTLEFINGER могут быть использованы для идентификации экземпляра(ов) биометрического типа, относящихся к пальцам.

Примечание – Подтип ЗБИ БиоАПИ соответствует «CBEFF_BDB_biometric_subtype» (ЕСФОБД_ББД_биометрический_подтип) в ИСО/МЭК 19785-1.

7.59.6 Если задана одна или более позиций бита с BioAPI_BIR_SUBTYPE_THUMB по BioAPI_BIR_SUBTYPE_LITTLEFINGER, то должна быть задана одна из двух позиций бита или обе позиции битов BioAPI_BIR_SUBTYPE_LEFT_MASK и BioAPI_BIR_SUBTYPE_RIGHT_MASK.

Если задан только бит в первой позиции, то значение подтипа обозначает один или более пальцев левой руки. Если задан только бит в последней позиции, то значение подтипа обозначает один или более паль-

цев правой руки. Если заданы биты в обеих позициях, то значение обозначает один или более пальцев обеих рук (один и тот же палец на обеих руках).

Примечание — Невозможно обозначить, к примеру, набор отпечатков пальцев, состоящий из указательного пальца левой руки и среднего пальца правой руки.

7.59.7 Если значение с набором битов нескольких пальцев присутствует в заголовке ЗБИ (или использовано в качестве входного параметра функции БиоАПИ, осуществляющей захват), то не обязательно, чтобы все указанные экземпляры пальцев действительно присутствовали в ББД ЗБИ (или были действительно захвачены). Однако рекомендуется, чтобы все обозначенные экземпляры пальцев присутствовали в ББД, кроме случая отсутствия пальца или подобных исключительных ситуаций. Таким же образом, если в поле подтипа ЗБИ обозначено несколько пальцев, то допускается, чтобы ББД ЗБИ включал данные о дополнительном пальце, если у субъекта шесть пальцев на руке.

7.59.8 Если ни одна позиция бита с BioAPI_BIR_SUBTYPE_THUMB по BioAPI_BIR_SUBTYPE_LITTLEFINGER не была задана, то позиции битов BioAPI_BIR_SUBTYPE_LEFT_MASK и BioAPI_BIR_SUBTYPE_RIGHT_MASK могут быть использованы для идентификации экземпляра биометрического типа, для которого существует один правый экземпляр и один левый экземпляр (например, радужная оболочка глаза), или идентификации биометрического типа, для которого существует лишь один экземпляр (например, лицо).

7.59.9 Данные рисунка вен, полученные из изображения одного или более пальцев, идентифицируются установкой бита в позиции 7 в ноль, установкой битов в одной или двух позициях 0 и 1, и установкой битов в одной или более позициях 3—7. Данные рисунка вен, полученные из других источников (запястье, ладонь или тыльная сторона ладони), идентифицируются установкой бита в позиции 7 в единицу, установкой битов в одной или двух позициях 0 и 1 и установкой одного из битов в позициях 3—5 соответствующим образом.

Примечание 1 – ЗБИ, которые не были созданы ПБУ БиоАПИ, но были преобразованы из другого формата данных, и для которых информация о подтипе недоступна, могут использовать значение NO VALUE AVAILABLE.

Примечание 2 – Подтип ЗБИ БиоАПИ соответствует «CBEFF_BDB_biometric_subtype» (ЕСФОБД_ББД_биометрический_подтип) в ИСО/МЭК 19785-1.

Примечание 3 – Данная структура, в первую очередь, используется в рамках заголовка ЗБИ; однако она также используется как входной параметр для функций, захватывающих биометрические данные. Значение BioAPI_NO_

SUBTYPE_AVAILABLE используется в заголовке ЗБИ, когда информация о подтипе недоступна. Значение BioAPI_NO_SUBTYPE_AVAILABLE также используется как параметр функции, когда приложение позволяет ЗБИ определять подтип, который должен быть захвачен.

7.59.10 Не рекомендуется использование позиции бита BioAPI_BIR_SUBTYPE_MULTIPLE, когда номер использующейся версии БиоАПИ 2.1.

7.60 Тип BioAPI_BSP_SCHEMA (БиоАПИ 2.1)

7.60.1 Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

7.60.2 Данный тип включает в себя информацию о ПБУ, содержащуюся в реестре компонентов.

```
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;
```

7.60.3 Определения

BSPUuid — УУИД ПБУ.

BSPDescription — строка с нулевым символом на конце, содержащая текстовое описание ПБУ.

Path — указатель на строку с нулевым символом на конце, содержащую путь к файлу, содержащему исполняемый код ПБУ, включая название файла. Путь к файлу может быть записан в виде адреса страницы URL. Символьная строка должна содержать символы, закодированные в формате UTF-8 в соответствии с ИСО/МЭК 10646 (приложение D).

Примечание – Если BioAPI_BSP_SCHEMA используется в вызове функции, компонент, получающий вызов, выделяет память для элемента схемы *Path* (путь к файлу), а вызывающий компонент освобождает память.

SpecVersion — номер редакции и номер поправки данной редакции спецификации БиоАПИ, для которой был разработан ПБУ.

ProductVersion — строка версии программного обеспечения ПБУ.

Vendor — строка с нулевым символом на конце, содержащая название изготовителя ПБУ.

BSPSupportedFormats — указатель на структуру BioAPI_BIR_BIOMETRIC_DATA_FORMAT, определяющую поддерживаемые форматы ББД.

NumSupportedFormats — число поддерживаемых форматов, содержащихся в BSPSupportedFormats.

FactorMask — маска, указывающая биометрические типы, поддерживаемые ПБУ.

Operations — маска, указывающая операции, поддерживаемые ПБУ.

Options — маска, указывающая опции, поддерживаемые ПБУ.

PayloadPolicy — пороговое значение (минимальное значение вероятности ложного совпадения), используемое для принятия решения о выдаче полезной информации после успешной верификации.

MaxPayloadSize — максимальный размер полезной информации (в байтах), которую может принять ПБУ.

DefaultVerifyTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для функции верификации ***BioAPI_Verify*** в случае, когда время ожидания не определено приложением.

DefaultIdentifyTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для функции идентификации ***BioAPI_Identify*** и ***BioAPI_IdentifyMatch*** в случае, когда время ожидания не определено приложением.

DefaultCaptureTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для функции захвата ***BioAPI_Capture*** в случае, когда время ожидания не определено приложением.

DefaultEnrollTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для функции регистрации

BioAPI_Enroll в случае, когда время ожидания не определено приложением.

DefaultCalibrateTimeout — заданное по умолчанию значение времени ожидания в миллисекундах, используемое ПБУ для операций калибровки датчика в случае, когда время ожидания не определено приложением.

MaxBSPDbSize — максимальный размер управляемой ПБУ базы данных ЗБИ.

Примечание 1 – Применяется только в том случае, когда ПБУ способен непосредственно управлять отдельным модулем архива.

Примечание 2 – Нулевое значение означает, что информация о размере базы данных не должна быть предоставлена по следующим трем причинам.

a) базы данных не поддерживаются,

b) существует возможность управления несколькими модулями (непосредственно или с использованием интерфейса ПБФ), каждый из которых может иметь различную максимальную длину, и информация о данном модуле будет предоставлена как часть уведомления о подключении (часть схемы модуля), или

c) поддерживается один модуль архива, но в данном параметре информация не предоставлена — она будет доступна в уведомлении о подключении.

MaxIdentify — максимальное число людей, поддерживаемых функцией идентификации. Значение FFFFFFFF указывает на отсутствие ограничения.

MaxNumEnrollInstances — максимальное число разных экземпляров, для которых ПБУ может создать контрольный шаблон в течение одной операции регистрации. Данная информация может быть полезной для приложения, использующего опцию ГИП, управляемого приложением.

HostingEndpointIRI — интернационализированный идентификатор ресурса (ИИР), идентифицирующий структуру, реестр компонентов которой содержит регистрацию ПБУ. Данный параметр должен быть проигнорирован инфраструктурами, соответствующими настоящему стандарту, и приложение должно установить его на пустой указатель. Это необходимо для поддержки взаимодействующих стандартов, которые могут определить порядок использования идентичных ПБУ, присутствующих на нескольких компьютерах, приложением, которое работает на том же или другом компьютере.

BSPAccessUUID — УУИД, уникальный в рамках области видимости приложения, с помощью которого приложение может сослаться на ПБУ вместо использования УУИД ПБУ. Данный параметр должен быть проигнорирован инфраструктурами, соответствующими настоящему стандарту, и приложение может ему задать любое значение УУИД. Это необходимо для поддержки взаимодействующих стандартов, которые могут

определить порядок использования идентичных ПБУ, присутствующих на нескольких компьютерах, приложением, которое работает на том же или другом компьютере.

Примечание – «BSPAccess_UUID» и «HostingendpointIRI» являются частью определения типа BioAPI_BSP_SCHEMA, но не являются частью информации о схеме ПБУ, хранящейся в реестре компонентов (см. таблицу 3).

7.60.4 Более подробное описание приведенных элементов и порядка записи информации ПБУ в реестр компонентов БиоАПИ приведено в 10.1.2 и 10.2.1.

7.61 Тип BioAPI_GUI_BITMAP (БиоАПИ 2.1)

7.61.1 Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

7.61.2 Данная структура представляет собой графическое изображение для его отображения приложением.

```
typedef struct bioapi_gui_bitmap {
  BioAPI_BIR_SUBTYPE_MASK SubtypeMask;
  /* Подтип (или подтипы) образца, представленного на изображении */
  uint32_t Width;
  /* Ширина изображения в точках (число точек в горизонтальной линии) */
  uint32_t Height;
  /* Высота изображения в точках (число горизонтальных линий) */
  BioAPI_DATA Bitmap;
} BioAPI_GUI_BITMAP;
```

7.61.3 Определения

Bitmap — множество 8-битовых полутоновых точек (где «00» — черный и «FF» — белый), в котором данные считываются слева направо и сверху вниз следующим образом (таблица 1-1).

Таблица 1-1 — Формат битовой карты графического интерфейса пользователя

Позиция байта	Значение	Описание
0	Строка 0, пиксель 0	Первый пиксель первой строки
1	Строка 0, пиксель 1	Второй пиксель первой строки
...	...	...

Окончание табл. 1-1

Позиция байта	Значение	Описание
Ширина — 1	Строка 0, пиксель (ширина — 1)	Последний пикеель первой строки
Ширина	Строка 1, пиксель 0	Первый пиксель второй строки
...	...	...
(Ширина*Высота) — 1	Строка (ширина — 1), пикеель (высота — 1)	Последняя строка, последний пиксель

Примечание — Данная структура используется вместе с опцией ГИП, управляемого приложением. Описание BioAPI_GUI_STATE_HANDLER и BioAPI_GUI_STREAMING_HANDLER приведено в 7.71.

7.62 Тип BioAPI_GUI_ENROLL_TYPE (БиоАПИ 2.1)

Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

Тип BioAPI_GUI_ENROLL_TYPE является типом регистрации ПБУ и определяется таким образом:

```
typedef uint32_t BioAPI_GUI_ENROLL_TYPE;
  #define BioAPI_GUI ENROLL_TYPE TEST_VERIFY
  (0x00000001)
  #define BioAPI_GUI ENROLL_TYPE MULTIPLE_CAPTURE
  (0x00000002)
```

Тип регистрации ПБУ указывает схему подопераций, представленных ПБУ в течение операции регистрации.

Позиция бита BioAPI_GUI ENROLL_TYPE TEST_VERIFY указывает (если задана) на то, что ПБУ поддерживает тест-верификацию при регистрации. В течение операции регистрации приложение получает обратные вызовы уведомления о событии изменения состояния ГИП для двух подопераций захвата, обратные вызовы уведомления о событии изменения состояния ГИП для подоперации обработки, обратные вызовы уведомления о событии изменения состояния ГИП для подоперации верификации — сравнения между «обработанной» ЗБИ и соответствующим шаблоном.

Позиция бита BioAPI_GUI ENROLL_TYPE TEST_VERIFY указывает (если задана) на то, что ПБУ поддерживает несколько подопераций захвата при регистрации. В течение операции регистрации приложение полу-

чает обратные вызовы уведомления о событии изменения состояния ГИП для нескольких подопераций захвата, за которыми следуют обратные вызовы уведомления о событии изменения состояния ГИП для подоперации создания шаблона.

Две позиции бита не могут быть заданы одновременно.

Примечание – Данное ограничение введено из-за того, что приложению для поддержки сочетания Тестировать-Верифицировать и Многократный-Захват потребуется больше информации о временных рамках подопераций для выбора разметки экрана в процессе регистрации.

7.63 Тип BioAPI_GUI_BITMAP_ARRAY (БиоАПИ 2.1)

Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

Данный тип BioAPI_GUI_BITMAP_ARRAY определяется следующим образом:

```
typedef struct bioapi_gui_bitmap_array {
  uint32_t NumberOfMembers;
  BioAPI_GUI_BITMAP *Bitmaps;
  /* Указатель на массив BioAPI_GUI_BITMAPs */
} BioAPI_GUI_BITMAP_ARRAY;
```

7.64 Тип BioAPI_BIR_SUBTYPE_MASK (БиоАПИ2.1)

Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

Данный тип BioAPI_GUI_SUBTYPE_MASK определяется следующим образом:

```
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)
```

Примечание 1 — Значения VEIN не присутствуют в версии БиоАПИ 2.0.

Позиции битов BioAPI_BIR_SUBTYPE_LEFT_BIT и BioAPI_BIR_SUBTYPE_RIGHT_BIT могут быть использованы для идентификации экземпляра биометрической модальности, для которой есть один «левый» экземпляр и один «правый» экземпляр.

Примечание 2 — Геометрия радужной оболочки глаза и руки является примером таких модальностей.

Другие позиции битов (с BioAPI_BIR_SUBTYPE_LEFT_THUMB_BIT по BioAPI_BIR_SUBTYPE_RIGHT_LITTLEFINGER_BIT) могут быть использованы для идентификации экземпляров биометрической модальности, касающейся пальцев.

Примечание 3 — Отпечаток пальца является примером такой модальности.

Значение ноль (позиция бита не задана) может быть использовано с биометрическими модальностями с единственным экземпляром.

Примечание 4 — Подпись является примером такой модальности.

7.65 Тип BioAPI_GUI_EVENT_SUBSCRIPTION (БиоАПИ 2.1)

Данный подраздел применяется только при использовании номера версии БиоАПИ 2.1.

Данный тип хранит информацию о существующей именованной подписке на события ГИП. Он идентифицирует приложение-подписчик и именованную подписку и указывает, какие типы событий ГИП находятся в области действий подписки. Данный тип предназначен для использования в функции ***BioAPI_QueryGUIEventSubscriptions.***

Именованная подписка на событие ГИП создана вызовом ***BioAPI_SubscribeToGUIEvents,*** определяющим непустой УУИД подписки на событие ГИП. Инфраструктура вызывает обработчики события, определенные именованной подписке, для уведомления о событиях ГИП, генерируемых ПБУ, которые перенаправлены на данную именованную подписку (см. 8.3.7), и для уведомления о событиях ГИП, генерируемых приложением (см. 8.3.3, 8.3.4, 8.3.5), которые направлены на данную именованную подписку.

```
typedef struct _bioapi_gui_event_subscription {
  const uint8_t *SubscriberEndpointIRI;
  BioAPI_UUID GUIEventSubscriptionUuid;
  BioAPI_BOOL GUISelectEventSubscribed;
  BioAPI_BOOL GUIStateEventSubscribed;
  BioAPI_BOOL GUIProgressEventSubscribed;
} BioAPI_GUI_EVENT_SUBSCRIPTION;
```

SubscriberEndpointIRI — ИИР — см. RFC 3987 — (первоначально предоставлен инфраструктурой), который идентифицирует приложение, создавшее именованную подписку на событие ГИП. Параметр должен быть установлен на пустой указатель, если приложение-подписчик то же, что и текущее приложение.

GUIEventSubscriptionUuid — УУИД (первоначально предоставлен приложением-подписчиком), который идентифицирует именованную подписку на событие ГИП.

GUISelectEventSubscribed — указывает на то, находятся ли события выбора ГИП в области действий подписки (адрес обратного вызова, установленный на непустой указатель, первоначально предоставлен приложением-подписчиком для обработчика события выбора ГИП).

GUIStateEventSubscribed — указывает на то, находятся ли события изменения состояния ГИП в области действий подписки (адрес обратного вызова, установленный на непустой указатель, первоначально предоставлен приложением-подписчиком для обработчика события изменения состояния ГИП).

GUIProgressEventSubscribed — определяет то, находятся ли события выполнения ГИП в области действий подписки (адрес обратного вызова, установленный на непустой указатель, первоначально предоставлен приложением-подписчиком для обработчика события выполнения ГИП).

7.66 Тип BioAPI_GUI_MOMENT (БиоАПИ 2.1)

Данный пункт применяется только при использовании версии БиоАПИ 2.1.

Перечень моментов, в которых:

a) событие выбора ГИП может быть сгенерировано в отношении цикла подоперации, или

b) событие изменения состояния или выполнения может быть сгенерировано в отношении подоперации.

```
typedef uint32_t BioAPI_GUI_MOMENT;
#define BioAPI_GUI_MOMENT_BEFORE_START (1)
#define BioAPI_GUI_MOMENT_DURING (2)
#define BioAPI_GUI_MOMENT_AFTER_END (3)
```

Значения BioAPI_GUI_MOMENT_BEFORE_START и BioAPI_GUI_MOMENT_AFTER_END могут появиться во всех типах уведомлений о событиях ГИП, в то время как значение BioAPI_GUI_MOMENT_DURING может появиться только в уведомлениях о событии выполнения ГИП.

Если значение BioAPI_GUI_MOMENT_BEFORE_START встречается в уведомлении о событии выбора ГИП, то это означает, что ПБУ готов запустить новый цикл подоперации. Цикл подоперации заключается в единственном выполнении последовательности подоперации, составляющих операцию (см. 7.68). Число циклов подоперации (успешных или нет), выполнение которых может быть затребовано приложением в рамках операции, не ограничено. Однако все полученные данные в течение каждого цикла должны быть отвергнуты с запуском последующего цикла, поэтому результатом операции (включая возвращаемое значение и полученные данные) всегда является результат ее последнего (или единственного) цикла подоперации (кроме случая отмены).

Если значение BioAPI_GUI_MOMENT_AFTER_END встречается в обратном вызове уведомления о событии выбора ГИП, то это означает, что текущий цикл подоперации завершен. Данное значение не указывает на то, успешно ли выполнение цикла подоперации или нет, так как информация предоставляется отдельным параметром уведомления о событии выбора ГИП, содержащего значение результата цикла подоперации.

Если значение BioAPI_GUI_MOMENT_BEFORE_START встречается в обратном вызове уведомления о событии изменения состояния ГИП, то это означает, что ПБУ готов запустить новую подоперацию в рамках текущего цикла подоперации.

Если значение BioAPI_GUI_MOMENT_AFTER_END встречается в обратном вызове уведомления о событии изменения состояния ГИП, то это означает, что текущая подоперация, выполняемая ПБУ, завершена.

Данное значение не указывает на то, успешно ли выполнение подоперации или нет, так как информация предоставляется отдельным параметром уведомления о событии изменения состояния ГИП, содержащего значение результата цикла подопераций.

Если значение BioAPI_GUI_MOMENT_BEFORE_START встречается в обратном вызове уведомления о событии выполнения ГИП, то это означает, что ПБУ готов запустить новую подоперацию в рамках текущего цикла подопераций. Событие выполнения ГИП с данным моментом может быть излишним, если также произведено событие изменения состояния ГИП, и может быть исключено. Однако подобное событие выполнения ГИП применимо в тех операциях (таких как идентификация), которые не производят событие изменения состояния ГИП.

Если значение BioAPI_GUI_MOMENT_DURING встречается в обратном вызове уведомления о событии выполнения ГИП, то это означает, что подоперация находится в процессе выполнения.

Если значение BioAPI_GUI_MOMENT_AFTER_END встречается в обратном вызове уведомления о событии выполнения ГИП, то это означает, что текущая подоперация, выполняемая ПБУ, завершена. Событие выполнения ГИП с данным моментом может быть излишним, если также произведено событие изменения состояния ГИП, и может быть исключено. Однако подобное событие выполнения ГИП применимо в тех операциях (таких как идентификация), которые не производят событие изменения состояния ГИП.

Примечание – Данный тип используется с опцией ГИП, управляемой приложением (см. 7.71).

7.67 Тип BioAPI_GUI_PROGRESS (БиоАПИ 2.1)

Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

Значение в диапазоне от 0 до 100 означает процент выполнения (завершения) подоперации, предназначенный для отображения пользователю или оператору.

```
typedef uint8_t BioAPI_GUI_PROGRESS;
```

Примечание – Используется с опцией ГИП, управляемого приложением. См. описание BioAPI_GUI_PROGRESS_EVENT в 7.71.

7.68 Тип BioAPI_GUI_OPERATION (БиоАПИ 2.1)

Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

Перечень всех функций (операций) интерфейса ПБУ, в процессе которых ПБУ может сгенерировать событие ГИП.

```
typedef uint8_t BioAPI_GUI_OPERATION;
  #define BioAPI_GUI_OPERATION_CAPTURE (1)
  #define BioAPI_GUI_OPERATION_PROCESS (2)
  #define BioAPI_GUI_OPERATION_CREATETEMPLATE (3)
  #define BioAPI_GUI_OPERATION_VERIFYMATCH (4)
  #define BioAPI_GUI_OPERATION_IDENTIFYMATCH (5)
  #define BioAPI_GUI_OPERATION_VERIFY (6)
  #define BioAPI_GUI_OPERATION_IDENTIFY (7)
  #define BioAPI_GUI_OPERATION_ENROLL (8)
```

Значения данного типа встречаются в уведомлениях о событии ГИП и идентифицируют тип функции интерфейса ПБУ, который выполнялся, когда ПБУ сгенерировал событие ГИП, и на который ссылается событие ГИП.

В таблице 1-2 определяется соответствие между функциями и операциями интерфейса ПБУ.

Таблица 1-2 — Функции и операции интерфейса ПБУ

Функции интерфейса ПБУ	Значение BioAPI_GUI_OPERATION	Общее наименование операции
BioSPI_Capture	BioAPI_GUI_OPERATION_CAPTURE	Операция захвата
BioSPI_Process	BioAPI_GUI_OPERATION_PROCESS	Операция обработки
BioSPI_Create-Template	BioAPI_GUI_OPERATION_CREATETEMPLATE	Операция создания шаблона
BioSPI_Verify-Match	BioAPI_GUI_OPERATION_VERIFYMATCH	Операция верификации и сопоставления
BioSPI_Identify-Match	BioAPI_GUI_OPERATION_IDENTIFYMATCH	Операция идентификации и сопоставления
BioSPI_Verify	BioAPI_GUI_OPERATION_VERIFY	Операция верификации

Окончание таблицы 1-2

Функции интерфейса ПБУ	Значение BioAPI_GUI_OPERATION	Общее наименование операции
BioSPI_Identify	BioAPI_GUI_OPERATION_ IDENTIFY	Операция идентификации
BioSPI_Enroll	BioAPI_GUI_OPERATION_ ENROLL	Операция регистрации

В таблице 1-3 определяется, какое из событий ГИП может быть сгенерировано ПБУ в процессе выполнения каждого типа операции.

Таблица 1-3 — События ГИП, генерируемые в каждой операции

Операция	События выбора ГИП	События изменения состояния ГИП	События обработки ГИП
Операция захвата	X	X	X
Операция обработки			X
Операция создания шаблона			X
Операция верификации и сопоставления			X
Операция идентификации и сопоставления			X
Операция верификации	X	X	X
Операция идентификации	X	X	X
Операция регистрации	X	X	X

7.69 Тип BioAPI_GUI_RESPONSE (БиоАПИ 2.1)

Данный подраздел применяется только при использовании версии БиоАПИ2.1

Перечень возможных действий, которые должны быть выполнены ПБУ после того как обратный вызов уведомления о событии ГИП, произведенный ПБУ, возвращает управление ПБУ. Приложение задает выходящему параметру обратного вызова (Response) значение данного типа перед возвратом от уведомления обратного вызова.

```
typedef uint8_t BioAPI_GUI_RESPONSE;
  #define BioAPI_GUI_RESPONSE_DEFAULT (0)
  #define BioAPI_GUI_RESPONSE_OP_COMPLETE (1)
  #define BioAPI_GUI_RESPONSE_OP_CANCEL (2)
  #define BioAPI_GUI_RESPONSE_CYCLE_START (3)
  #define BioAPI_GUI_RESPONSE_CYCLE_RESTART (4)
  #define BioAPI_GUI_RESPONSE_SUBOP_START (5)
  #define BioAPI_GUI_RESPONSE_SUBOP_NEXT (6)
  #define BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE (7)
  #define BioAPI_GUI_RESPONSE_PROGRESS_ABORT (8)
  #define BioAPI_GUI_RESPONSE_RECAPTURE (9)
```

Значение BioAPI_GUI_RESPONSE_OP_COMPLETE может быть возвращено только в качестве ответа обратному вызову уведомления о событии выбора ГИП со значением момента BioAPI_GUI_MOMENT_AFTER_END. Оно означает то, что ПБУ должен считать операцию завершенной и должен инициировать вызов первоначальной функции для возвращения кода результата текущего цикла подопераций (который может быть либо BioAPI_OK, либо кодом ошибки). После данного ответа приложения в отношении той же операции обратные вызовы уведомления о событии ГИП не ожидаются.

Значение BioAPI_GUI_RESPONSE_OP_CANCEL может быть возвращено в качестве ответа на любой обратный вызов уведомления о событии ГИП. Оно означает то, что ПБУ должен отменить всю операцию и инициировать вызов первоначальной функции для возвращения кода ошибки. После данного ответа приложения в отношении той же операции обратные вызовы уведомления о событии ГИП не ожидаются.

Значение BioAPI_GUI_RESPONSE_CYCLE_START может быть возвращено только в качестве ответа обратному вызову уведомления о событии выбора ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START. Оно означает то, что ПБУ должен запустить цикл подопераций.

Значение BioAPI_GUI_RESPONSE_CYCLE_RESTART может быть возвращено только в качестве ответа обратному вызову уведомления о событии выбора ГИП со значением момента BioAPI_GUI_ MOMENT_AFTER_END или в качестве ответа обратному вызову уведомления о событии изменения состояния или выполнения с любым значением момента. Оно означает то, что ПБУ должен отвергнуть все данные, полученные в течение текущего цикла подопераций, и должен запустить новый цикл подопераций. После данного ответа приложения от ПБУ ожидается обратный вызов уведомления о событии выбора ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START.

Значение BioAPI_GUI_RESPONSE_SUBOP_START может быть возвращено только в качестве ответа обратному вызову уведомления о событии изменения состояния ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START. Оно означает то, что ПБУ должен запустить подоперацию.

Значение BioAPI_GUI_RESPONSE_SUBOP_NEXT может быть возвращено только в качестве ответа обратному вызову уведомления о событии изменения состояния ГИП со значением момента BioAPI_GUI_MOMENT_AFTER_END. Оно означает то, что ПБУ должен либо выполнить следующую подоперацию в текущем цикле подопераций (если существуют еще подоперации, которые необходимо выполнить), либо выйти из цикла подопераций (если все подоперации цикла выполнены). После данного ответа приложения от ПБУ ожидается обратный вызов уведомления о событии изменения состояния ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START или обратный вызов уведомления о событии выбора со значением момента BioAPI_GUI_MOMENT_AFTER_END (соответственно).

Значение BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE может быть возвращено только в качестве ответа на обратный вызов уведомления о событии выполнения ГИП. Оно означает то, что ПБУ должен продолжить выполнение подоперации.

Значение BioAPI_GUI_RESPONSE_PROGRESS_ABORT может быть возвращено только в качестве ответа на обратный вызов уведомления о событии выполнения ГИП. Оно означает то, что ПБУ должен отменить подоперацию и выдать код ошибки. После данного ответа приложения от ПБУ ожидается обратный вызов уведомления о событии изменения ГИП со значением момента BioAPI_GUI_MOMENT_AFTER_END.

Значение BioAPI_GUI_RESPONSE_RECAPTURE может быть возвращено только в качестве ответа на обратный вызов уведомления о событии выполнения ГИП со значением момента BioAPI_GUI_MOMENT_AFTER_END. Оно означает то, что ПБУ должен отвергнуть один из полученных в текущем цикле подопераций образцов (возможно, отвергая и любой контрольный шаблон и любой обработанный образец, созданные из отвергаемого образца) и выполнить другую подоперацию захвата для создания нового образца. После данного ответа приложения от ПБУ ожидается обратный вызов уведомления о событии изменения состояния ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START (для подоперации захвата с целью регистрации). По завершении подоперации захвата ПБУ должен выполнить любую другую подоперацию, необходимую для завершения текущего цикла подопераций.

Значение BioAPI_GUI_RESPONSE_DEFAULT должно пониматься как одно из следующих значений, которое относится к особой ситуации, при которой оно возвращено: BioAPI_GUI_RESPONSE_CYCLE_START, BioAPI_GUI_RESPONSE_SUBOP_START, BioAPI_GUI_RESPONSE_PROCESS_CONTINUE, BioAPI_GUI_RESPONSE_SUBOP_NEXT, или BioAPI_GUI_RESPONSE_OP_COMPLETE.

Если приложение возвращает не то ответное значение, которое допустимо в каждой отдельной ситуации, то ПБУ должен отменить операцию и вызвать первоначальную функцию для возвращения кода ошибки. В отношении той же операции обратные вызовы уведомления о событии ГИП не ожидаются.

В таблице 1-4 сведена информация о совместимости между типами событий ГИП, значениями момента и ответами.

Таблица 1-4 — Ответы ГИП и события ГИП

Ответ приложения	Событие выбора ГИП (перед запуском)	Событие изменения состояния ГИП (перед запуском)	Событие выполнения ГИП (перед запуском)	Событие выполнения ГИП (в процессе)	Событие выполнения ГИП (после завершения)	Событие изменения состояния ГИП (после завершения)	Событие выбора ГИП (после завершения)
BioAPI_GUI_RESPONSE_DEFAULT	X	X	X	X	X	X	X
BioAPI_GUI_RESPONSE_CYCLE_START	X						
BioAPI_GUI_RESPONSE_SUBOP_START		X					
BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE			X	X	X		
BioAPI_GUI_RESPONSE_SUBOP_NEXT						X	
BioAPI_GUI_RESPONSE_OP_COMPLETE							X

Окончание таблицы 1-4

Ответ приложения	Событие выбора ГИП (перед запуском)	Событие изменения состояния ГИП (перед запуском)	Событие выполнения ГИП (перед запуском)	Событие выполнения ГИП (в процессе)	Событие выполнения ГИП (после завершения)	Событие изменения состояния ГИП (после завершения)	Событие выбора ГИП (после завершения)
BioAPI_GUI_RESPONSE_PROGRESS_ABORT			X	X	X		
BioAPI_GUI_RESPONSE_RECAPTURE						X	
BioAPI_GUI_RESPONSE_CYCLE_RESTART		X	X	X	X	X	X
BioAPI_GUI_RESPONSE_OP_CANCEL	X	X	X	X	X	X	X

7.70 Тип BioAPI_GUI_SUBOPERATION (БиоАПИ 2.1)

Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

Перечень возможных типов подопераций, выполняемых ПБУ в процессе операции, о которых может быть сообщено приложению в уведомлении о событии ГИП.

```
typedef uint8_t BioAPI_GUI_SUBOPERATION;
  #define BioAPI_GUI_SUBOPERATION_CAPTURE (1)
  #define BioAPI_GUI_SUBOPERATION_PROCESS (2)
  #define BioAPI_GUI_SUBOPERATION_CREATETEMPLATE (3)
  #define BioAPI_GUI_SUBOPERATION_VERIFYMATCH (4)
  #define BioAPI_GUI_SUBOPERATION_IDENTIFYMATCH (5)
```

Значения данного типа встречаются в уведомлениях о событии ГИП и идентифицируют тип подоперации текущей операции, для которой сгенерировано событие ГИП.

В таблице 1-5 показаны типы подопераций (и их число), которые выполняются ПБУ в процессе каждого типа операции (по порядку). Для операций BioAPI_GUI_OPERATION_CAPTURE, BioAPI_GUI_OPERATION_VERIFY, BioAPI_GUI_OPERATION_IDENTIFY и BioAPI_GUI_OPERATION_ENROLL число подопераций соответствует полному циклу подопераций (см. 7.66), и в нем не учитываются возможные повторные захваты, которые совершаются в процессе цикла (см. 7.69).

Таблица I-5 — Подоперации, выполняемые в каждой операции

Операции	Подоперации
BioAPI_GUI_OPERATION_CAPTURE с целью верификации или идентификации	**BioAPI_GUI_SUBOPERATION_CAPTURE** (одна) **BioAPI_GUI_SUBOPERATION_PROCESS** (ни одной или одна)
BioAPI_GUI_OPERATION_CAPTURE с целью регистрации (если тип регистрации тестировать-верифицировать)	**BioAPI_GUI_SUBOPERATION_CAPTURE** (две) **BioAPI_GUI_SUBOPERATION_CREATETEMPLATE** (одна) **BioAPI_GUI_SUBOPERATION_PROCESS** (одна) **BioAPI_GUI_SUBOPERATION_VERIFYMATCH** (одна)
BioAPI_GUI_OPERATION_CAPTURE с целью регистрации (если тип регистрации многократный-захват)	**BioAPI_GUI_SUBOPERATION_CAPTURE** (несколько) **BioAPI_GUI_SUBOPERATION_CREATETEMPLATE** (одна)
BioAPI_GUI_OPERATION_PROCESS	**BioAPI_GUI_SUBOPERATION_PROCESS** (одна)
BioAPI_GUI_OPERATION_CREATETEMPLATE	**BioAPI_GUI_SUBOPERATION_CREATETEMPLATE** (одна)
BioAPI_GUI_OPERATION_VERIFYMATCH	**BioAPI_GUI_SUBOPERATION_VERIFYMATCH** (одна)
BioAPI_GUI_OPERATION_IDENTIFYMATCH	**BioAPI_GUI_SUBOPERATION_IDENTIFYMATCH** (одна)
BioAPI_GUI_OPERATION_VERIFY	**BioAPI_GUI_SUBOPERATION_CAPTURE** (одна) **BioAPI_GUI_SUBOPERATION_PROCESS** (одна) **BioAPI_GUI_SUBOPERATION_VERIFYMATCH** (одна)

Окончание таблицы 1-5

Операции	Подоперации
BioAPI_GUI_OPERATION_IDENTIFY	**BioAPI_GUI_SUBOPERATION_CAPTURE** (одна) **BioAPI_GUI_SUBOPERATION_PROCESS** (одна) **BioAPI_GUI_SUBOPERATION_IDENTIFYMATCH** (одна)
BioAPI_GUI_OPERATION_ENROLL (если тип регистрации тестировать-верифицировать)	**BioAPI_GUI_SUBOPERATION_CAPTURE** (две) **BioAPI_GUI_SUBOPERATION_CREATETEMPLATE** (одна) **BioAPI_GUI_SUBOPERATION_PROCESS** (одна) **BioAPI_GUI_SUBOPERATION_VERIFYMATCH** (одна)
BioAPI_GUI_OPERATION_ENROLL (если тип регистрации многократный-захват)	**BioAPI_GUI_SUBOPERATION_CAPTURE** (несколько) **BioAPI_GUI_SUBOPERATION_CREATETEMPLATE** (одна)

7.71 События ГИП

Данный подраздел применяется только при использовании версии БиоАПИ 2.1.

7.71.1 BioAPI_GUI_SELECT_EVENT_HANDLER (БиоАПИ 2.1)

7.71.1.1 Функция обратного вызова

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_SELECT_EVENT_
HANDLER)
    (const BioAPI_UUID *BSPUuid,
    BioAPI_UNIT_ID UnitID,
    const BioAPI_HANDLE *BSPHandle,
    BioAPI_GUI_ENROLL_TYPE EnrollType,
    const void *GUISelectEventHandlerCtx,
    BioAPI_GUI_OPERATION Operation,
    BioAPI_GUI_MOMENT Moment,
    BioAPI_RETURN ResultCode,
    int32_t MaxNumEnrollSamples,
    BioAPI_BIR_SUBTYPE_MASK SelectableInstances,
    BioAPI_BIR_SUBTYPE_MASK *SelectedInstances,
    BioAPI_BIR_SUBTYPE_MASK CapturedInstances,
    const uint8_t *Text,
    BioAPI_GUI_RESPONSE *Response);
```

7.71.1.2 Описание

Это тип указателя на функцию для функции обработчика событий ГИП приложения, который обрабатывает обратные вызовы уведомления о событии выбора ГИП, исходящие от ПБУ через инфраструктуру БиоАПИ. Для получения уведомлений о событии выбора ГИП приложение должно зарегистрировать функцию обратного вызова типа BioAPI_GUI_SELECT_EVENT_HANDLER путем предоставления адреса обратного вызова функции вместе с адресом контекста в вызове BioAPI_SubscribeToGUIEvents (см. 8.3.8).

Инфраструктура производит вызов функции приложения данного типа каждый раз, когда получает входной обратный вызов функции BioSPI_GUI_SELECT_EVENT_HANDLER, которую объявляет для ПБУ. Параметры обратного вызова (кроме GUISelectEventHandlerCtx) должны быть заданы из параметров входящего обратного вызова с теми же именами; параметр GUISelectEventHandlerCtx должен быть задан из адреса контекста выбора ГИП, предоставленного первоначально приложением в его вызове BioAPI_SubscribeToGUIEvents; адрес обратного вызова должен быть задан из адреса обратного вызова выбора ГИП, предоставленного первоначально приложением в вызове BioAPI_SubscribeToGUIEvents.

ПБУ может генерировать события выбора ГИП только в процессе выполнения вызова BioAPI_Capture, BioAPI_Verify, BioAPI_Identify или BioAPI_Enroll. Событие выбора ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START генерируется перед запуском каждого цикла подопераций, а событие выбора ГИП со значением момента BioAPI_GUI_MOMENT_AFTER_END генерируется после завершения каждого цикла подопераций (см. 7.66). Основное назначение уведомления о событии выбора ГИП, которое генерируется перед запуском цикла подопераций:

1) сообщить приложению о том, что новый цикл подопераций готов к запуску;

2) разрешить приложению выбрать экземпляр(ы) для захвата (где возможность подобного выбора поддерживается ПБУ и соответствует используемой биометрической модальности).

Основное назначение уведомления о событии выбора ГИП, которое генерируется после завершения цикла подопераций: сообщить приложению о результатах цикла подопераций. Приложение должно ответить каждому уведомлению о событии выбора ГИП путем указания на то, должен ли цикл подопераций запуститься, должна ли операция считаться полностью завершенной, должен ли запуститься новый цикл подопераций или операция должна быть отменена (см. 7.69).

Если приложение определяет один или больше экземпляров в параметре Subtype вызова первоначальной функции (BioAPI_Capture, BioAPI_Enroll, BioAPI_Verify или BioAPI_Identify), и этот параметр поддерживается ПБУ, то параметр SelectableInstances задается исходя из значения параметра Subtype.

Функция обработки события выбора ГИП и любые другие вызванные функции из этой функции не должны вызывать (непосредственно или косвенно) ни одну функцию БиоАПИ. Возврат значения, отличающегося от BioAPI_OK, приводит к немедленному возврату вызванной функции к вызывающему коду с передачей ему данного значения в качестве кода ошибки.

7.71.1.3 Параметры

BSPUuid (входной) — УУИД, идентифицирующий ПБУ, связанный с событием ГИП.

UnitID (входной) — ИД модуля датчика ПБУ, связанного с событием ГИП.

BSPHandle (входной) — дескриптор присоединенной сессии ПБУ, связанный с событием ГИП.

EnrollType (входной) — тип регистрации ПБУ. Данный параметр имеет значение, только если *Operation* является BioAPI_GUI_OPERATION_ENROLL.

GUISelectEventHandlerCtx (входной) — контекстный адрес, первоначально предоставленный приложением-подписчиком как часть подписки на событие ГИП.

Operation (входной) — значение, которое указывает на тип операции, для которой сгенерировано событие ГИП. Допустимы только следующие значения:

- BioAPI_GUI_OPERATION_CAPTURE;
- BioAPI_GUI_OPERATION_VERIFY;
- BioAPI_GUI_OPERATION_IDENTIFY;
- BioAPI_GUI_OPERATION_ENROLL.

Moment (входной) — значение, которое указывает на то — сгенерировано ли событие ГИП перед запуском цикла подопераций или после завершения цикла подопераций. Допустимы только следующие значения:

- BioAPI_GUI_MOMENT_BEFORE_START;
- BioAPI_GUI_MOMENT_AFTER_END.

ResultCode (входной) — данный параметр является значащим только тогда, когда значением параметра Moment является BioAPI_GUI_MOMENT_AFTER_END, в противном случае ему должно быть задано значение, равное нулю. Если параметр является значащим, то он должен содержать код результата только что завершенного цикла подопераций. Код результата может быть либо BioAPI_OK, либо код ошибки БиоАПИ.

MaxNumEnrollSamples (входной) — данный параметр является значащим только тогда, когда значением параметра Moment является BioAPI_GUI_MOMENT_BEFORE_START. Если параметр является значащим, то он должен содержать максимальное число образцов, которые ПБУ может захватить в течение цикла подопераций, который скоро запустится.

SelectableInstances (входной) — данный параметр является значащим только тогда, когда значением параметра Moment является BioAPI_GUI_MOMENT_BEFORE_START. Если параметр является значащим, то он должен содержать битовую маску, которая определяет какой экземпляр(ы) биометрической модальности может(гут) быть выбран(ы) приложением для захвата в течение цикла подопераций, который скоро запустится. Некоторые ПБУ поддерживают выбор нескольких экземпляров модальности, потому что они могут создать в единственной подоперации захвата единственную ЗБИ, содержащую данные, касающиеся нескольких экземпляров.

SelectedInstances (выходной) — данный параметр является значащим только тогда, когда значением параметра Moment является BioAPI_GUI_MOMENT_BEFORE_START. Если параметр является значащим, то он должен содержать битовую маску, которая определяет какой(ие) экземпляр(ы) биометрической модальности выбран(ы) приложением для захвата в течение цикла подопераций, который скоро запустится.

CapturedInstances (входной) — данный параметр является значащим только тогда, когда значением параметра Moment является BioAPI_GUI_MOMENT_AFTER_END, в противном случае ему должно быть задано значение, равное нулю. Если параметр является значащим, то он должен содержать битовую маску, которая определяет какой(ие) экземпляр(ы) успешно захвачены ПБУ в течение цикла подопераций, который только что завершился.

Text (входной) — текстовое сообщение, состоящее из строк символов, закодированных в формате UTF-8 в соответствии с ИСО/МЭК 10646, предназначенное для демонстрации пользователю или оператору. Содержание сообщения и язык, в котором оно выражено, не стандартизированы.

Response (выходной) — значение, определяющее ответ приложения на уведомление о событии выбора ГИП (см. 7.69).

Примечание – См. также С 7.

7.71.1.4 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

7.71.1.5 Ошибки:

BioAPIERR_USER_CANCELLED

BioAPIERR_FUNCTION_FAILED

7.71.2 BioAPI_GUI_STATE_EVENT_HANDLER (БиоАПИ 2.1)

7.71.2.1 Функция обратного вызова

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STATE_EVENT_
HANDLER)
(const BioAPI_UUID *BSPUuid,
BioAPI_UNIT_ID UnitID,
const BioAPI_HANDLE *BSPHandle,
const void *GUIStateEventHandlerCtx,
BioAPI_GUI_OPERATION Operation,
BioAPI_GUI_SUBOPERATION Suboperation,
```

BioAPI_BIR_PURPOSE Purpose,
BioAPI_GUI_MOMENT Moment,
BioAPI_RETURN ResuilCode,
int32_t EnrollSampleIndex,
const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
const uint8_t *Text,
BioAPI_GUI_RESPONSE *Response,
int32_t *EnrollSampleIndexToRecapture);

7.71.2.2 Описание

Данный указатель указывает на функцию обработчика событий ГИП приложения, которая обрабатывает обратные вызовы уведомления о событиях изменения состояния ГИП, исходящие от ПБУ через инфраструктуру БиоАПИ. Для того чтобы получить уведомления о событии изменения состояния ГИП, приложение должно зарегистрировать функцию обратного вызова типа BioAPI_GUI_STATE_EVENT_HANDLER путем предоставления адреса обратного вызова функции вместе с контекстным адресом в вызове ***BioAPI_SubscribeToGUIEvents*** (см. 8.3.8).

Инфраструктура производит обратный вызов функции приложения данного типа каждый раз, когда получает входящий обратный вызов функции типа BioSPI_GUI_STATE_EVENT_HANDLER, которую она открывает ПБУ. Параметры обратного вызова (кроме *GUIStateEventHandlerCtx*) должны быть заданы из параметров входящего обратного вызова с теми же наименованиями; параметр *GUIStateEventHandlerCtx* должен быть задан из контекстного адреса изменения состояния ГИП, предоставленного приложением в его вызове ***BioAPI_SubscribeToGUIEvents;*** адрес обратного вызова должен быть задан из адреса обратного вызова изменения состояния ГИП, предоставленного приложением в вызове ***BioAPI_SubscribeToGUIEvents.***

ПБУ может генерировать события изменения состояния ГИП только в процессе выполнения вызова ***BioAPI_Capture, BioAPI_Verify, BioAPI_Identify*** или ***BioAPI_Enroll.*** Событие изменения состояния ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START генерируется перед запуском каждой подоперации, а событие изменения состояния ГИП со значением момента BioAPI_GUI_MOMENT_AFTER_END генерируется после завершения каждой подоперации (см. 7.66).

Основное назначение уведомления о событии изменения состояния ГИП, которое генерируется перед запуском подоперации: сообщить приложению о том, что определенная подоперация готова к запуску. Основное назначение уведомления о событии изменения состояния ГИП, кото-

рое генерируется после завершения подоперации: сообщить приложению о результатах подоперации.

Приложение должно ответить каждому уведомлению о событии изменения состояния ГИП путем указания на то, должна ли подоперация запуститься, должен ли цикл подопераций продолжиться следующей подоперацией, должен ли запуститься новый цикл подопераций или операция должна быть отменена (см. 7.69).

Функция обработки события выбора ГИП и любые другие вызванные функции из этой функции не должны вызывать (непосредственно или косвенно) ни одну функцию БиоАПИ.

Возврат значения, отличающегося от BioAPI_OK, приводит к немедленному возврату вызванной функции к вызывающему коду с передачей ему данного значения в качестве кода ошибки.

7.71.2.3 Параметры

BSPUuid (входной) — УУИД, идентифицирующий ПБУ, связанный с событием ГИП.

UnitID (входной) — ИД модуля датчика ПБУ, связанного с событием ГИП.

BSPHandle (входной) — дескриптор присоединенной сессии ПБУ, связанный с событием ГИП.

GUIStateEventHandlerCtx (входной) — контекстный адрес, первоначально предоставленный приложением-подписчиком как часть подписки на событие ГИП.

Operation (входной) — значение, которое указывает на тип операции, для которой сгенерировано событие ГИП. Допустимы только следующие значения:

- BioAPI_GUI_OPERATION_CAPTURE;
- BioAPI GUI_OPERATION_VERIFY;
- BioAPI_GUI_OPERATION_IDENTIFY;
- BioAPI_GUI_OPERATION_ENROLL.

Suboperation (входной) — значение, которое указывает на тип подоперации, для которой сгенерировано событие ГИП (подоперация, которая скоро запустится или только что завершилась). Подоперация должна быть совместима с операцией (см. таблицу 1-5).

Purpose (входной) — данный параметр является значащим только тогда, когда значением параметра Suboperation является BioAPI_GUI_SUBOPERATION_CAPTURE, в противном случае ему должно быть задано значение, равное нулю. Если параметр является значащим, то он должен указывать на назначение подоперации захвата (любое значение типа BioAPI_BIR_PURPOSE).

Moment (входной) — значение, которое указывает на то, сгенерировано ли событие ГИП перед запуском подоперации или после завершения подоперации. Допустимы только следующие значения:

- BioAPI_GUI_MOMENT_BEFORE_START;
- BioAPI_GUI_MOMENT_AFTER_END.

ResultCode (входной) — данный параметр является значащим только тогда, когда значением параметра Moment является BioAPI_GUI_MOMENT_AFTER_END, в противном случае ему должно быть задано значение, равное нулю. Если параметр является значащим, то он должен содержать код результата только что завершенного цикла подопераций. Код результата может быть либо BioAPI_OK, либо код ошибки БиоАПИ.

EnrollSampleIndex (входной) — данный параметр является значащим только тогда, когда значением параметра Suboperation является BioAPI_GUI_SUBOPERATION_CAPTURE, значением параметра Moment является BioAPI_GUI_MOMENT_BEFORE_START, и образец для регистрации скоро будет захвачен. В противном случае ему должно быть задано значение, равное нулю. Если параметр является значащим, то он должен содержать индекс (целое число от 1 и более) захватываемого для регистрации образца в подоперации захвата, которая скоро запустится.

Bitmaps (входной) — последовательность битовых изображений, содержащая изображение(я) образцов, полученных в процессе подоперации, которые предназначены для демонстрации пользователю или оператору. Данная последовательность в основном не содержит битовые изображения или содержит одно битовое изображение, но может содержать несколько битовых изображений, если у ПБУ есть возможность захватывать несколько экземпляров биометрической модальности в течение одной операции захвата.

Text (входной) — текстовое сообщение, состоящее из строк символов, закодированных в формате UTF-8 в соответствии с ИСО/МЭК 10646, предназначенное для демонстрации пользователю или оператору. Содержание сообщения и язык, в котором оно выражено, не стандартизированы.

Response (выходной) — значение, определяющее ответ приложения на уведомление о событии выбора ГИП (см. 7.69).

EnrollSampleIndexToRecapture (выходной) — данный параметр является значащим только тогда, когда значением параметра Suboperation является BioAPI_GUI_SUBOPERATION_CREATETEMPLATE, значением параметра Moment является BioAPI_GUI_MOMENT_AFTER_END, и значением параметра *Response* является BioAPI_GUI_RESPONSE_

RECAPTURE. Если параметр является значащим, то он должен содержать индекс (положительное целое число), полученный ранее для регистрации образца, перезахват которого требует приложение.

Пример — Если ПБУ предоставляет функции BioAPI_GUI_SELECT_EVENT_HANDLER тип регистрации многократный-захват, то в каждой операции регистрации функция может захватить три биометрических образца. ПБУ генерирует события изменения состояния перед запуском и после завершения каждой подоперации захвата (когда EnrollSampleIndex заданы значения 1, 2 и 3), за которыми следует событие изменения состояния ГИП перед запуском и после завершения подоперации создания шаблона. Если приложение получает уведомление о событии изменения состояния ГИП перед запуском подоперации создания шаблона, то приложение спрашивает у пользователя, какой из образцов должен быть перемещен. Пользователь выбирает второй образец после просмотра битовых изображений на экране. Приложение задает параметру Response значение BioAPI_GUI_RESPONSE_RECAPTURE, EnrollSampleIndexToRecapture значение 2 и выходит из функции обратного вызова. После этого ПБУ генерирует другое событие изменения состояния ГИП перед запуском и после завершения новой подоперации захвата, когда EnrollSampleIndex задано значение 2, и, наконец, другое событие изменения состояния ГИП перед запуском и после завершения новой подоперации создания шаблона.

Примечание – См. также С.7.

7.71.2.4 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

7.71.2.5 Ошибки:

BioAPIERR_USER_CANCELLED

BioAPIERR_FUNCTION_FAILED

7.71.3 BioAPI_GUI_PROGRESS_EVENT_HANDLER (БиоАПИ 2.1)

7.71.3.1 Функция обратного вызова

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_PROGRESS_EVENT_
HANDLER)
(const BioAPI_UUID *BSPUuid,
BioAPI_UNIT_ID UnitID,
const BioAPI_HANDLE *BSPHandle,
const void *GUIProgressEventHandlerCtx,
BioAPI_GUI_OPERATION Operation,
```

BioAPI_GUI_SUBOPERATION Suboperation,
BioAPI_BIR_PURPOSE Purpose,
BioAPI_GUI_MOMENT Moment,
uint8_t SuboperationProgress,
const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
const uint8_t *Text,
BioAPI_GUI_RESPONSE *Response);

7.71.3.2 Описание

Данный указатель указывает на функцию обработчика событий ГИП приложения, которая обрабатывает обратные вызовы уведомления о событиях выполнения ГИП, исходящие от ПБУ через инфраструктуру БиоАПИ.

Для того чтобы получить уведомления о событии выполнения ГИП приложение должно зарегистрировать функцию обратного вызова типа BioAPI_GUI_PROGRESS_EVENT_HANDLER путем предоставления адреса обратного вызова функции вместе с контекстным адресом в вызове ***BioAPI_SubscribeToGUIEvents*** (см. 8.3.8).

Инфраструктура производит обратный вызов функции приложения данного типа каждый раз, когда получает входящий обратный вызов функции типа BioSPI_GUI_PROGRESS_EVENT_HANDLER, которую она открывает ПБУ. Параметры обратного вызова (кроме *GUIProgressEventHandlerCtx*) должны быть заданы из параметров входящего обратного вызова с теми же наименованиями; параметр *GUIProgressEventHandlerCtx* должен быть задан из контекстного адреса ГИП, предоставленного приложением в его вызове ***BioAPI_SubscribeToGUIEvents;*** адрес обратного вызова должен быть задан из адреса обратного вызова выполнения ГИП, предоставленного приложением в вызове ***BioAPI_SubscribeToGUIEvents.***

ПБУ может генерировать события выполнения ГИП только в процессе выполнения вызова ***BioAPI_Capture, BioAPI_Process, BioAPI_CreateTemplate, BioAPI_VerifyMatch, BioAPI_IdentifyMatch, BioAPI_Verify, BioAPI_Identify*** или ***BioAPI_Enroll.*** События могут быть сгенерированы в любое время, даже многократно в течение любой подоперации.

Основное назначение уведомления о событии выполнения ГИП: сообщить приложению о степени выполнения любой продолжительной подоперации в процентах (таких как подоперации захвата или идентификации-сопоставления) и отправить приложению потоковые данные (в виде набора битовых изображений), собранные датчиком. Один из возможных способов использования данной информации приложением заключается в демонстрации битовых изображений и индикатора выполнения пользова-

телю или оператору. Приложение должно отвечать каждому уведомлению о событии выполнения ГИП путем указания на то, должна ли подоперация продолжиться или быть прекращена (см. 7.69).

Функция обработки события выбора ГИП и любые другие вызванные функции из этой функции не должны вызывать (непосредственно или косвенно) ни одну функцию БиоАПИ.

Возврат значения, отличающегося от BioAPI_OK, приводит к немедленному возврату вызванной функции к вызывающему коду с передачей ему данного значения в качестве кода ошибки.

7.71.3.3 Параметры

BSPUuid (входной) — УУИД, идентифицирующий ПБУ, связанный с событием ГИП.

UnitID (входной) — ИД модуля датчика ПБУ, связанного с событием ГИП.

BSPHandle (входной) — дескриптор присоединенной сессии ПБУ, связанный с событием ГИП.

GUIProgressEventHandlerCtx (входной) — контекстный адрес, первоначально предоставленный приложением-подписчиком как часть подписки на событие ГИП.

Operation (входной) — значение, которое указывает на тип операции, для которой сгенерировано событие ГИП.

Suboperation (входной) — значение, которое указывает на тип подоперации, для которой сгенерировано событие ГИП. Подоперация должна быть совместима с операцией (см. таблицу 1-5).

Purpose (входной) — данный параметр является значащим только тогда, когда значением параметра *Suboperation* является BioAPI_GUI_SUBOPERATION_CAPTURE, в противном случае ему должно быть задано значение, равное нулю. Если параметр является значащим, то он должен указывать на назначение подоперации захвата (любое значение типа BioAPI_BIR_PURPOSE).

Moment (входной) — значение, которое указывает на то, сгенерировано ли событие ГИП перед запуском подоперации, в течение подоперации или после завершения подоперации.

SuboperationProgress (входной) — степень выполнения подоперации в процентах.

Bitmaps (входной) — последовательность битовых изображений, содержащая изображение(я) образцов, полученных в процессе подоперации, которые предназначены для демонстрации пользователю или оператору. Данная последовательность в основном не содержит битовые изображения или содержит одно битовое изображение, но может содержать несколько битовых изображений, если у ПБУ есть возможность захваты-

вать несколько экземпляров биометрической модальности в течение одной операции захвата.

Text (входной) — текстовое сообщение, состоящее из строк символов, закодированных в формате UTF-8 в соответствии с ИСО/МЭК 10646, предназначенное для демонстрации пользователю или оператору. Содержание сообщения и язык, в котором оно выражено, не стандартизированы.

Response (выходной) — значение, определяющее ответ приложения на уведомление о событии выбора ГИП (см. 7.69).

Примечание – См также С.7.

7.71.3.4 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

7.71.3.5 Ошибки:

BioAPIERR_USER_CANCELLED

BioAPIERR_FUNCTION_FAILED

7.72 BioAPI_ERROR_INFO (БиоАПИ 2.1)

Данный подраздел применяется только при использовании БиоАПИ версии 2.1.

Данная структура используется функцией ***BioAPI_GetLastErrorInfo*** и содержит информацию о последнем типе ошибки, инициировавшем функцию БиоАПИ возвратить код ошибки:

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

Значением элемента *ErrorCode* является то же самое, что было возвращено последним вызовом функции БиоАПИ, возвратившим код ошибки. Значения остальных элементов не стандартизированы».

Подпункт 8.1.1.1. Второй абзац изложить в новой редакции:

«Любой вызов функции ***BioAPI_Init,*** если предыдущий вызов функции ***BioAPI_Init*** или ***BioAPI_InitEndpoint*** (только в БиоАПИ 2.1) не был закончен соответствующим вызовом функции ***BioAPI_Terminate,*** будет обработан следующим образом: инфраструктура БиоАПИ выдаст ответ BioAPI_OK (только в том случае, если номер версии инфраструктуры совместим с номером версии, который был предоставлен предыдущим вызовом функции ***BioAPI_Init*** или ***BioAPI_InitEndpoint)*** или

BioAPI_ERR_INCOMPATIBLE_VERSION, но не будет инициализирована повторно. Счетчик числа удачных вызовов ***BioAPI_Init*** или ***BioAPI_ InitEndpoint*** будет сохраняться инфраструктурой и не завершится до тех пор, пока число соответствующих вызовов ***BioAPI_Terminate*** не достигнет определенного значения».

Подпункт 8.1.2.1. Второй абзац изложить в новой редакции:

«Данная функция может быть вызвана только в том случае, если функция ***BioAPI_Init*** или ***BioAPI_InitEndpoint*** (только в БиоАПИ 2.1), для которой еще не была вызвана соответствующая функция, ранее была вызвана успешно, по крайней мере, один раз. Функция ***BioAPI_Terminate*** не должна вызываться до успешного выполнения функции ***BioAPI_Init*** или ***BioAPI_InitEndpoint*** (например, вызовы, которые не были закончены связанными вызовами ***BioAPI_Terminate***)».

Подпункт 8.1.4.1. Заменить абзац: «Данная функция может быть вызвана только в том случае, если был сделан, по крайней мере, один вызов функции ***BioAPI_Init,*** для которого еще не был сделан соответствующий вызов функции ***BioAPI_Terminate***» на «Данная функция может быть вызвана только в том случае, если был сделан, по крайней мере, один вызов функции ***BioAPI_Init*** или ***BioAPI_InitEndpoint*** (только в БиоАПИ 2.1), для которого еще не было произведено соответствующего вызова функции ***BioAPI_Terminate***».

Подпункт 8.1.5.1. Первый абзац после слов «разрешение всех событий» дополнить словами: «,кроме тех, которые были деактивированы последним вызовом ***BioAPI_EncibleEventNotifications*** (в БиоАПИ 2.1)»;

седьмой абзац. Заменить слова: «Если ПБУ загружен путем вызова функции ***BioAPI_BSPLoad***, он должен для каждого существующего модуля БиоАПИ немедленно вызвать событие «установка» на «Когда инфраструктура БиоАПИ вызывает функцию ***BioSPI_BSPLoad,*** она для каждого существующего модуля БиоАПИ от ПБУ получает уведомление о событии «установка». Если биометрическое приложение предоставило в вызове функции ***BioAPI_BSPLoad*** обработчик события и не деактивировало уведомления о событии «установка», то инфраструктура, в свою очередь, произведет обратный вызов обработчика событий приложения. Если используется БиоАПИ версии 2.1, то уведомления о событии установки могут быть деактивированы при помощи вызова функции ***BioAPI_EnableEventNotiflcations*** (см. примечание в 7.27)»;

предпоследний абзац изложить в новой редакции:

«Данная функция может быть вызвана только в том случае, если был выполнен, по крайней мере, один вызов функции ***BioAPI_Init*** или ***BioAPI_ InitEndpoint*** (только в БиоАПИ 2.1), для которого еще не было произведено соответствующего вызова функции ***BioAPI_Terminate.*** Функция

BioAPI_BSPAttach может быть вызвана несколько раз после выполнения функции ***BioAPI_BSPLoad***».

Подпункт 8.1.7.1 дополнить абзацами и примечанием:

«При использовании БиоАПИ версии 2.1, если модуль БиоАПИ, используемый в присоединенной сессии, становится недействительным, то и присоединенная сессия должна быть также признана недействительной. Когда присоединенная сессия признана недействительной, то единственными функциями, определяющими дескриптор данной присоединенной сессии, которые можно вызвать, являются функции ***BioAPI_ BSPDetach*** и ***BioAPI_GetBIRFromHandle***. Все остальные функции должны возвращать BioAPIERR_INVALID_ATTACH_SESSION.

Если ПБУ поддерживает события, то он может создать событие «отключения», когда модуль БиоАПИ признан недействительным. При использовании БиоАПИ версии 2.1, если приложение установило обработчик события и не деактивировало уведомления о событии «отключение», то оно получает уведомление о событии «отключение» и может сделать вывод, что все присоединенные сессии (если существуют), использующие отключенный модуль БиоАПИ, на текущий момент недействительны. В дальнейшем приложение может либо удалить эти присоединенные сессии с помощью вызова ***BioAPI_BSPDetach,*** либо быть готовым к получению ошибки BioAPIERR_INVALID_ATTACH_SESSION от последующего вызова функции, а потом вызвать ***BioAPI_BSPDetach.***

Примечание – Если модуль БиоАПИ был косвенно выбран для присоединенной сессии (либо посредством использования функции ***BioAPI_DONT_CARE,*** либо посредством невключения категории модуля в список, предоставленный в качестве входного параметра ***BioAPI_BSPAttach***), то приложение может не знать, какие присоединенные сессии использовали отключенный модуль БиоАПИ и, таким образом, не иметь возможности удалить эти присоединенные сессии без предварительного получения ошибок BioAPIERR_INVALID_ATTACH_SESSION при последующих вызовах функций».

Подпункт 8.1.7.2. Последний абзац. Заменить обозначение: «BioAPIERR_FRAMEWORK_INVALID_BSP_HANDLE» на «BioAPI_ INVALID_BSP_HANDLE».

Подпункт 8.1.10.1. Заменить абзац: «Данная функция может быть вызвана только в том случае, если был сделан, по крайней мере, один вызов функции ***BioAPI_Init,*** для которого еще не был сделан соответствующий вызов функции ***BioAPI_Terminate***» на «Данная функция может быть вызвана только в том случае, если был сделан, по крайней мере, один вызов функции ***BioAPI_Init*** или ***BioAPI_InitEndpoint*** (только в БиоАПИ 2.1), для которого еще не был сделан соответствующий вызов функции ***BioAPI_Terminate***».

Подраздел 8.1 дополнить пунктами — 8.1.13—8.1.18:

«8.1.13 Функция BioAPI_Control (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

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

8.1.13.1 Описание

Данная функция посылает управляющие данные от приложения модулю БиоАПИ и получает обратно данные состояния или рабочие данные. Содержание параметра *ControlCode* посылаемых (входных) данных и получаемых (выходных) данных должно быть определено в спецификации на интерфейс для данного модуля БиоАПИ (или связанного ИПФ в том случае, если он присутствует).

Данная функция выделяет область памяти, достаточную для размещения выходных данных, которые должны быть возвращены приложению, заполняет блок данными и записывает в поля *Length* и *Data* структуры *OutputData* размер и адрес блока памяти (соответственно).

Область памяти, возвращенная при вызове функции БиоАПИ, должна быть освобождена приложением путем вызова функции ***BioAPI_Free*** (8.7.2).

8.1.13.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

UnitId (входной) — ИД модуля БиоАПИ.

ControlCode (входной) — код функции в вызываемом модуле БиоАПИ.

InputData (входной) — адрес и длина буфера данных, которые должны быть посланы модулю БиоАПИ в соответствии с полученным *ControlCode*.

OutputData (выходной) — указатель на структуру BioAPI_DATA. На выходе она должна содержать адрес и длину буфера данных, содержащего данные, полученные от модуля БиоАПИ после обработки функции, указанной в *ControlCode*. Если функция не выделила область памяти, то адрес должен быть установлен на пустой указатель, а длина буфера задана равной нулю.

8.1.13.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.13.4 Ошибки

BioAPIERR_BIOAPI_UNIT_NOT_INSERTED
BioAPIERR_INVALID_UNIT_ID
BioAPIERR_UNIT_IN_USE
BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.14 Функция BioAPI_Transform (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_Transform
 (BioAPI_HANDLE BSPHandle,
 const BioAPI_UUID *OperationUUID,
 const BioAPI_INPUT_BIR *InputBIRs,
 uint32_t NumberOfInputBIRs,
 BioAPI_BIR_HANDLE  **OutputBIRs,
 uint32_t *NumberOfOutputBIRs);
```

8.1.14.1 Описание

Данная функция трансформирует одну или несколько ЗБИ, предоставленных в качестве входного параметра, в одну или несколько ЗБИ, предоставленных в качестве выходного параметра. Выполняемая трансформация определяется параметром *OperationUUID*.

Настоящий стандарт не устанавливает никаких стандартных значений для параметра *OperationUUID* и не устанавливает никаких определенных трансформаций. Предполагается, что значения *OperationUUID* и их семантика будут установлены либо поставщиком определенного ПБУ, либо в дополнительных спецификациях (таких как сведения о приложении).

Данная функция выполняет действия в следующем порядке:

a) выполняет трансформацию, определенную параметром *OperationUUID*, используя ЗБИ, предоставленные в качестве входного параметра, и создавая одну или несколько выходных ЗБИ, как требуется для определенной трансформации;

b) выделяет область памяти, достаточную для размещения массива элементов типа BioAPI_BIR_HANDLE с числом элементов, равным числу созданных выходных ЗБИ в перечислении a);

c) заполняет массив информацией о дескрипторах ЗБИ, созданных в качестве выходного параметра в перечислении a); и

d) возвращает адрес массива в параметре *OutputBIRs* и размер массива в параметре *NumberOfOutputBIRs*.

Область памяти, возвращаемая вызовом функции БиоАПИ, должна быть освобождена приложением с помощью функции ***BioAPI_Free***

(см. 8.7.2), а все существующие в массиве ***OutputBIRs*** дескрипторы ЗБИ должны быть удалены путем вызова функции ***BioAPI_FreeBIRHandle.***

8.1.14.2 Параметры

BSPHandle (входной) — дескриптор присоединенного ПБУ.

OperationUUID (входной) — УУИД, определяющий трансформацию, выполняемую ПБУ.

InputBIRs (входной) — массив ЗБИ (содержащий одну или несколько ЗБИ), предоставленных в качестве входного параметра трансформации.

NumberOfInputBIRs (входной) — число ЗБИ в массиве InputBIRs.

OutputBIRs (выходной) — указатель на адрес массива, состоящий из элементов типа BioAPI_BIR_HANDLE и содержащий дескрипторы выходных ЗБИ, созданные трансформацией.

NumberOfOutputBIRs (выходной) — указатель на число элементов массива OutputBIRs.

8.1.14.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на отсутствие ошибки. Все остальные значения описывают тип ошибки.

8.1.14.4 Ошибки

BioAPIERR_BIOAPI_UNIT_NOT_INSERTED
BioAPIERR_UNIT_IN_USE
BioAPIERR_TRANSFORMATION_NOT_SUPPORTED
BioAPIERR_INVALID_BSP_HANDLE

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.15 Функция BioAPI_InitEndpoint (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_InitEndpoint
  (BioAPI_VERSION Version,
  const uint8_t *LocalEndpointIRI);
```

8.1.15.1 Описание

Данная функция выполняет те же действия, что и функция ***BioAPI_Init.***

Приложения, соответствующие настоящему стандарту, должны либо не вызывать функцию, либо (если вызывают ее) должны установить данный параметр *LocalEndpointIRI* на пустой указатель. Функция предоставлена для поддержки стандартов межсетевого обмена.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБУ.

8.1.15.2 Параметры

Version (входной) — номер редакции и номер поправки спецификации БиоАПИ, с которой совместимо биометрическое приложение.

LocalEndpointIRI — данный параметр должен быть проигнорирован инфраструктурами, соответствующими этому разделу настоящего стандарта, и должен быть установлен на пустой указатель приложением. Он предоставлен для поддержки стандартов межсетевого обмена.

8.1.15.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.1.15.4 Ошибки

BioAPIERR_INCOMPATIBLE_VERSION

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.16 Функция BioAPI_LinkToEndpoint (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_LinkToEndpoint
  (const uint8_t *SlaveEndpointIRI);
```

8.1.16.1 Описание

Данная функция не выполняет никаких действий и предоставлена для измененных спецификаций, содержащихся в стандартах межсетевого обмена, которые определяют применение инфраструктур БиоАПИ, установленных на нескольких компьютерах, из приложения, работающего на том же или другом компьютере. В этих стандартах данная функция может быть использована для соединения приложения с инфраструктурой, работающей на другом компьютере, таким образом, предоставляя приложению доступ к ПБУ, управляемому этой же инфраструктурой.

Приложения, соответствующие настоящему стандарту, должны либо не вызывать функцию, либо (если вызывают ее) должны установить данный параметр *SlaveEndpointIRI* на пустой указатель. Функция предоставлена для поддержки стандартов межсетевого обмена.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБУ.

8.1.16.2 Параметры

SlaveEndpointIRI — данный параметр должен быть проигнорирован инфраструктурами, соответствующими этому разделу настоящего стандарта, и должен быть установлен на пустой указатель приложением. Он предоставлен для поддержки стандартов межсетевого обмена.

8.1.16.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.1.16.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.17 Функция BioAPI_UnlinkFromEndpoint (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_UnlinkFromEndpoint
  (const uint8_t *SlaveEndpointIRI);
```

8.1.17.1 Описание

Данная функция не выполняет никаких действий и предоставлена для измененных спецификаций, содержащихся в стандартах межсетевого обмена, которые определяют применение инфраструктур БиоАПИ, установленных на нескольких компьютерах из приложения, работающего на том же или другом компьютере. В этих стандартах данная функция может быть использована для разрушения соединения между приложением и инфраструктурой.

Приложения должны либо не вызывать функцию, либо (если вызывают ее) должны установить данный параметр *SlaveEndpointIRI* на пустой указатель.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБУ.

8.1.17.2 Параметры

SlaveEndpointIRI — данный параметр должен быть проигнорирован инфраструктурами, соответствующими этому разделу настоящего стандарта, и должен быть установлен на пустой указатель приложением. Он предоставлен для поддержки стандартов межсетевого обмена.

8.1.17.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.1.17.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.1.18 Функция BioAPI_EnumFrameworks (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_EnumFrameworks
  (BioAPI_FRAMEWORK_SCHEMA **FwSchemaArray,
   unit32_t *NumberOfElements);
```

8.1.18.1 Описание

Данная функция схожа с функцией ***BioAPI_GetFrameworkInfo,*** но позволяет получить информацию о нескольких инфраструктурах БиоАПИ

при использовании спецификации, предоставленной стандартом межсетевого обмена.

В спецификации, предоставленной в настоящем стандарте, данная функция возвращает информацию об одной инфраструктуре и, следовательно, аналогична ***BioAPI_GetFrameworkInfo*** (хотя и с разными параметрами).

Данная функция предоставляет информацию обо всех инфраструктурах БиоАПИ, которые на текущий момент видны приложению (только одна, за исключением использования стандарта межсетевого обмена). Данная функция выполняет действия в следующем порядке:

a) выделяет область памяти, достаточную для размещения массива элементов типа BioAPI_FRAMEWORK_SCHEMA с числом элементов, равным числу видимых инфраструктур;

b) заполняет массив схемами инфраструктур для всех видимых инфраструктур; и

c) возвращает адрес массива в параметре *FwSchemaArray* и число элементов массива в параметре *NumberOfElements*.

Данная функция должна вызываться только в том случае, если был произведен хотя бы один вызов ***BioAPI_Init*** или ***BioAPI_InitEndpoint,*** для которого еще не было произведено соответствующего вызова ***BioAPI_Terminate.***

Данная функция обрабатывается в Инфраструктуре БиоАПИ и не передается ни одному ПБУ.

Область памяти, содержащая массив, должна быть освобождена приложением с помощью вызова функции ***BioAPI_Free*** (см. 8.7.2), в том случае если она больше не используется приложением. Области памяти, указанные элементами *Path* и *HostingEndpoint* в рамках каждого элемента массива, также должны быть освобождены приложением с помощью вызова функции ***BioAPI_Free,*** в том случае, если они больше не используются приложением.

8.1.18.2 Параметры

BFPSchemaArray (выходной) — указатель на адрес массива элементов типа BioAPI_FRAMEWORK_SCHEMA (распределенного инфраструктурой), содержащего информацию о схемах инфраструктур.

NumberOfElements (выходной) — указатель на число элементов массива (число инфраструктур, на текущий момент видимых приложению — только одна, за исключением использования стандарта межсетевого обмена).

8.1.18.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на ус-

пешное выполнение функции. Все остальные значения описывают тип ошибки.

8.1.18.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11».

Подпункт 8.3.1.1. Первый абзац дополнить словами: «Применение данной функции не рекомендуется при использовании БиоАПИ версии 2.1. В этом случае должна использоваться функция ***BioAPI_EnableEventNotifications»***.

Пункт 8.3.2. Заголовок изложить в новой редакции:

«Функция BioAPI_SetGUICallbacks (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный пункт применяется только при использовании версии БиоАПИ 2.0».

Подраздел 8.3 дополнить пунктами — 8.3.3—8.3.11:

«8.3.3 Функция BioAPI_NotifyGUIProgressEvent (БиоАПИ 2.1)

Данный пункт применяется только при использовании версии БиоАПИ 2.1.

```
BioAPI_RETURN BioAPI BioAPI_NotifyGUIProgressEvent
  (const uint8_t *SubscriberEndpointIRI,
  const BioAPI_UUID *GUIEventSubscriptionUuid,
  const BioAPI_UUID *BSPUuid,
  BioAPI_UNIT_ID UnitID,
  BioAPI_GUI_OPERATION Operation,
  BioAPI_GUI_SUBOPERATION Suboperation,
  BioAPI_BIR_PURPOSE Purpose,
  BioAPI_GUI_MOMENT Moment,
  uint8_t SuboperationProgress,
  const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
  const uint8_t *Text,
  BioAPI_GUI_RESPONSE *Response);
```

8.3.3.1 Описание

Данная функция дает возможности приложению требовать того, чтобы событие выполнения ГИП, сгенерированное приложением, было направлено в подписку определенного именованного события ГИП. Событие ГИП, переданное данной функции, называется событием ГИП, сгенерированным приложением.

При вызове данной функции инфраструктура должна просматривать все существующие именованные подписки на события ГИП, созданные приложением, идентифицированным *SubscriberEndpointIRI* (пустой указатель — текущее приложение), в поисках подписки, в которой адрес обратного вызова для события выполнения ГИП установлен на непус-

той указатель, а УУИД подписки на событие ГИП тот же, что и *GUIEventSubscriptionUuid*. При наличии подписки на сопоставление инфраструктура должна произвести обратный вызов обработчику события выполнения ГИП, определенного в подписке, задавая входные параметры обратного вызова из входных параметров вызова ***BioAPI_NotifyGUIProgressEvent*** с теми же наименованиями. После возврата из функции обратного вызова инфраструктура должна скопировать выходные параметры обратного вызова в выходные параметры вызова ***BioAPI_NotifyGUIProgressEvent*** с теми же наименованиями.

Если совпадающих именованных подписок на событие ГИП не существует, то функцией возвращается BioAPI_NO_GUI_EVENT_HANDLER.

Данная функция может быть вызвана только в том случае (для установленного УУИД ПБУ), если был произведен хотя бы один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого еще не был произведен соответствующий вызов ***BioAPI_BSPUnload.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

8.3.3.2 Параметры

SubscriberEndpointIRI — интернационализированный идентификатор ресурсов, идентифицирующий приложение, создавшее одну или больше подписок на событие ГИП. Параметр должен быть установлен на пустой указатель, если данное приложение является текущим приложением. Приложение может узнать об интернационализированных идентификаторах ресурсов конечной точки подписчика других приложений, использующих инфраструктуру (и создавших одну или больше именованных подписок на событие ГИП для установленного ПБУ) путем вызова ***BioAPI_QueryGUIEventSubscriptions.*** Если других приложений, использующих инфраструктуру, не существует, то ИИР конечной точки подписчика установлены на пустые указатели во всех именованных подписках, возвращенных ***BioAPI_QueryGUIEventSubscriptions.***

GUIEventSubscriptionUuid — УУИД, идентифицирующий именованную подписку на событие ГИП. Данный параметр не должен быть установлен на пустой указатель. В некоторых случаях приложение получает данный УУИД из информации, возвращенной предшествующим вызовом ***BioAPI_QueryGUIEventSubscriptions;*** в других случаях приложение предоставляет известный УУИД.

Другие параметры определяют событие выполнения ГИП, сгенерированное приложением.

8.3.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK

указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.3.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.4 Функция BioAPI_NotifyGUISelectEvent (БиоАПИ 2.1)

Данный пункт применяется только при использовании версии БиоАПИ 2.1.

```
BioAPI_RETURN BioAPI_BioAPI_NotifyGUISelectEvent
  (const uint8_t *SubscriberEndpointIRI,
  const BioAPI_UUID *GUIEventSubscriptionUuid,
  const BioAPI_UUID *BSPUuid,
  BioAPI_UNIT_ID UnitID,
  BioAPI_GUI_OPERATION Operation,
  BioAPI_GUI_MOMENT Moment,
  BioAPI_RETURN ResultCode,
  int32_t MaxNumEnrollSamples,
  BioAPI_BIR_SUBTYPE_MASK SelectableInstances,
  BioAPI_BIR_SUBTYPE_MASK *SelectedInstances,
  BioAPI_BIR_SUBTYPE_MASK CapturedInstances,
  const uint8_t *Text,
  BioAPI_GUI_RESPONSE *Response);
```

8.3.4.1 Описание

Данная функция дает возможности приложению требовать того, чтобы событие выбора ГИП, сгенерированное приложением, было привязано в подписку определенного именованного события ГИП. Событие ГИП, переданное данной функции, называется событием ГИП, сгенерированным приложением.

При вызове данной функции инфраструктура должна просматривать все существующие именованные подписки на события ГИП, созданные приложением, идентифицированным *SubscriberEndpointIRI* (пустой указатель — текущее приложение), в поисках подписки, в которой адрес обратного вызова для события выполнения ГИП установлен на непустой указатель, а УУИД подписки на событие ГИП тот же, что и *GUIEventSubscriptionUuid*. При наличии подписки на сопоставление инфраструктура должна произвести обратный вызов обработчику события выбора ГИП, определенного в подписке, задавая входные параметры обратного вызова из входных параметров вызова ***BioAPI_NotifyGUISelectEvent*** с теми же наименованиями. После возврата из функции обратного вызова инфраструктура должна скопировать выходные параметры обратного вызова в выходные параметры вызова ***BioAPI_NotifyGUISelectEvent*** с теми же наименованиями.

Если совпадающих именованных подписок на событие ГИП не существует, то функцией возвращается BioAPI_NO_GUI_EVENT_HANDLER.

Данная функция может быть вызвана только в том случае (для установленного УУИД ПБУ), если был произведен хотя бы один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого еще не был произведен соответствующий вызов ***BioAPI_BSPUnload.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

8.3.4.2 Параметры

SubscriberEndpointIRI (входной, дополнительный) — интернационализированный идентификатор ресурсов, идентифицирующий приложение, создавшее одну или больше подписок на событие ГИП. Параметр должен быть установлен на пустой указатель, если данное приложение является текущим приложением. Приложение может узнать об интернационализированных идентификаторах ресурсов конечной точки подписчика других приложений, использующих инфраструктуру (и создавших одну или больше именованных подписок на событие ГИП для установленного ПБУ) путем вызова ***BioAPI_QueryGUIEventSubscriptions.*** Если других приложений, использующих инфраструктуру, не существует, то ИИР конечной точки подписчика установлены на пустые указатели во всех именованных подписках, возвращенных ***BioAPI_QueryGUIEventSubscriptions.***

GUIEventSubscriptionUuid (входной) — УУИД, идентифицирующий именованную подписку на событие ГИП. Данный параметр не должен быть установлен на пустой указатель. В некоторых случаях приложение получает данный УУИД из информации, возвращенной предшествующим вызовом ***BioAPI_QueryGUIEventSubscriptions;*** в других случаях приложение предоставляет известный УУИД.

Другие параметры определяют событие выбора ГИП, сгенерированное приложением.

8.3.4.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.4.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.5 Функция BioAPI_NotifyGUISelectEvent (БиоАПИ 2.1)

Данный пункт применяется только при использовании версии БиоАПИ 2.1.

```
BioAPI_RETURN BioAPI BioAPI_NotifyGUIStateEvent
  (const uint8_t *SubsriberEndpointIRI,
  const BioAPI_UUID *GUIEventSubscriptionUuid,
  const BioAPI_UUID *BSPUuid,
  BioAPI_UNIT_ID UnitID,
  BioAPI_GUI_OPERATION Operation,
  BioAPI_GUI_SUBOPERATION Suboperation,
  BioAPI_BIR_PURPOSE Purpose,
  BioAPI_GUI_MOMENT Moment,
  BioAPI_RETURN ResultCode,
  int32_t EnrollSampleIndex,
  const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
  const uint8_t *Text,
  BioAPI_GUI_RESPONSE *Response,
  int32_t *EnrollSampleIndexToRecapture);
```

8.3.5.1 Описание

Данная функция дает возможности приложению требовать того, чтобы событие изменения состояния ГИП, сгенерированное приложением, было привязано к подписке определенного именованного события ГИП. Событие ГИП, переданное данной функции, называется событием ГИП, сгенерированным приложением.

При вызове данной функции инфраструктура должна просматривать все существующие именованные подписки на события ГИП, созданные приложением, идентифицированным *SubscriberEndpointIRI* (пустой указатель — текущее приложение), в поисках подписки, в которой адрес обратного вызова для события изменения состояния ГИП установлен на непустой указатель, а УУИД подписки на событие ГИП тот же, что и *GUIEventSubscriptionUuid*. При наличии подписки на сопоставление инфраструктура должна произвести обратный вызов обработчику события изменения состояния ГИП, определенного в подписке, задавая входные параметры обратного вызова из входных параметров вызова ***BioAPI_NotifyGUIStateEvent*** с теми же наименованиями. После возврата из функции обратного вызова инфраструктура должна скопировать выходные параметры обратного вызова в выходные параметры вызова ***BioAPI_NotifyGUIStateEvent*** с теми же наименованиями.

Если совпадающих именованных подписок на событие ГИП не существует, то функцией возвращается **BioAPI_NO_GUI_EVENT_HANDLER.**

Данная функция может быть вызвана только в том случае (для установленного УУИД ПБУ), если был произведен хотя бы один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого еще не был произведен соответствующий вызов ***BioAPI_BSPUnload.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

8.3.5.2 Параметры

SubscriberEndpointIRI (входной, дополнительный) — интернационализированный идентификатор ресурсов, идентифицирующий приложение, создавшее одну или больше подписок на событие ГИП. Параметр должен быть установлен на пустой указатель, если данное приложение является текущим приложением. Приложение может узнать об интернационализированных идентификаторах ресурсов конечной точки подписчика других приложений, использующих инфраструктуру (и создавших одну или больше именованных подписок на событие ГИП для установленного ПБУ) путем вызова ***BioAPI_QueryGUIEventSubscriptions.*** Если других приложений, использующих инфраструктуру, не существует, то ИИР конечной точки подписчика установлены на пустые указатели во всех именованных подписках, возвращенных ***BioAPI_QueryGUIEventSubscriptions.***

GUIEventSubscriptionUuid (входной) — УУИД, идентифицирующий именованную подписку на событие ГИП. Данный параметр не должен быть установлен на пустой указатель. В некоторых случаях приложение получает данный УУИД из информации, возвращенной предшествующим вызовом ***BioAPI_QueryGUIEventSubscriptions***; в других случаях приложение предоставляет известный УУИД.

Другие параметры определяют событие изменения состояния ГИП, сгенерированное приложением.

8.3.5.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.5.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.6 Функция BioAPI_QueryGUIEventSubscriptions (БиоАПИ 2.1)

Данный пункт применяется только при использовании версии БиоАПИ 2.1.

```
BioAPI_RETURN BioAPI BioAPI_QueryGUIEventSubscriptions
  (const BioAPI_UUID *BSPUuid,
  BioAPI_GUI_EVENT_SUBSCRIPTION**GUIEventSubscriptionList,
  uint32_t *NumberOfElements);
```

8.3.6.1 Описание

Данная функция предоставляет информацию обо всех существующих именованных подписках на события ГИП для установленного ПБУ. Именованной подпиской на событие ГИП называется та подписка, которая

создана путем вызова функции ***BioAPI_SubscribeToGUIEvents,*** определяющего УУИД подписки на событие ГИП, установленный на непустой указатель. Инфраструктура вызывает обработчики событий, определенные в именованной подписке, для уведомления событий ГИП, сгенерированных ПБУ, и перенаправленных в данную именованную подписку (см. 8.3.7) и для уведомления событий ГИП, сгенерированных приложением (см. 8.3.3, 8.3.4 и 8.3.5), направленных в данную именованную подписку.

Путем вызова данной функции приложение может узнать об интернационализированных идентификаторах ресурсов конечной точки подписчика других приложений, использующих инфраструктуру (где эта возможность поддерживается архитектурой реализации инфраструктуры БиоАПИ) и создавших именованные подписки на событие ГИП для установленного ПБУ. Если других приложений, использующих инфраструктуру, не существует, то ИИР конечной точки подписчика установлены на пустой указатель во всех именованных подписках, возвращенных данной функцией.

Приложение может использовать информацию, возвращенную вызовом данной функции (ИИР конечной точки подписчика, УУИД подписки на событие ГИП и флаги), в последующих вызовах функции ***BioAPI_RedirectGUIEvent*** (для требования того, чтобы определенные последующие события ГИП, сгенерированные ПБУ, перенаправить определенным именованным подпискам) или в последующих вызовах ***BioAPI_NotifyGUISelectEvent, BioAPI_NotifyGUIStateEvent*** и т. д. (для требования того, чтобы событие ГИП, сгенерированное приложением, направить определенным именованным подпискам).

Данная функция выполняет действия в следующем порядке:

a) выделяет область памяти, достаточную для размещения массива элементов типа BioAPI_GUI_EVENT_SUBSCRIPTION с числом элементов, равным числу существующих именованных подписок на событие ГИП для установленного ПБУ;

b) заполняет массив информацией об именованных подписках; и

c) возвращает адрес массива в параметре *GUIEventSubscriptionList* и число элементов массива в параметре *NumberOfElements*.

Данная функция может быть вызвана только в том случае (для установленного УУИД ПБУ), если был произведен хотя бы один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого еще не был произведен соответствующий вызов ***BioAPI_BSPUnload.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

Область памяти, содержащая массив, должна быть освобождена приложением с помощью вызова функции ***BioAPI_Free*** (см. 8.7.2), в том случае если он больше не нужен приложению.

8.3.6.2 Параметры

GUIEventSubscriptionList (выходной) — указатель на адрес массива элементов типа BioAPI_GUI_EVENT_SUBSCRIPTION (выделенного инфраструктурой), содержащего информацию об именованной подписке на событие ГИП.

NumberOfElements (выходной) — указатель на число элементов массива.

8.3.6.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.6.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.7 Функция BioAPI_RedirectGUIEvents (БиоАПИ 2.1)

Данный пункт применяется только при использовании номера версии БиоАПИ 2.1.

```
BioAPI_RETURN BioAPI BioAPI_RedirectGUIEvents
  (const uint8_t *SubscriberEndpointIRI,
  const BioAPI_UUID *GUIEventSubscriptionUuid,
  BioAPI_HANDLE BSPHandle,
  BioAPI_BOOL GUISelectEventRedirected,
  BioAPI_BOOL GUIStateEventRedirected,
  BioAPI_BOOL GUIProgressEventRedirected);
```

8.3.7.1 Описание

Данная функция создает «перенаправитель событий ГИП» для текущего приложения. Перенаправители событий ГИП поддерживаются инфраструктурой во время выполнения (обычно во внутренней таблице) и дают возможность приложению передавать обработку событиями ГИП, сгенерированными ПБУ, либо другому приложению, использующему инфраструктуру (где это поддерживается архитектурой реализации инфраструктуры), либо другому компоненту того же приложения.

Приложение, вызывающее данную функцию, должно предоставить ИИР конечной точки подписчика приложения, создавшего именованную подписку на событие ГИП (пустой указатель, если это то же приложение) и УУИД подписки на событие ГИП, как и информацию, определяющую область действия перенаправителя события ГИП (какая присоединенная сессия ПБУ и какой(ие) тип(ы) события(ий) ГИП подвер-

гаются воздействию). Все значения параметров, предоставленные в вызове функции, должны стать частью перенаправителя события ГИП, поддерживаемого инфраструктурой.

Каждый вызов данной функции создает новый перенаправитель событий ГИП и не изменяет, а также не заменяет существующий перенаправитель. Перенаправители событий ГИП, определяющие установленный дескриптор присоединенной сессии ПБУ, должны быть автоматически удалены инфраструктурой, когда присоединенная сессия удалена. В любое время приложение может запросить инфраструктуру об удалении существующего перенаправителя с помощью вызова ***BioAPI_UnredirectGUIEvents,*** предоставляя те же параметры, что были предоставлены в соответствующем вызове ***BioAPI_RedirectGUIEvents.***

Результатом многократного вызова данной функции, в любое время, не должны становиться многократные перенаправители событий выбора ГИП, многократные перенаправители событий изменения состояния ГИП или многократные перенаправители событий выполнения ГИП, созданные для одного и того же дескриптора ПБУ. Это гарантирует определенность в способе перенаправления события для любого входящего события ГИП.

Примечание – Предполагается недопустимость двух перенаправителей с одинаковыми параметрами.

Если инфраструктура получает вызов данной функции, который успешно создал перенаправитель события ГИП, то она в ответ должна вызвать функцию ***BioSPI_SubscribeToGUIEvents*** ПБУ только в том случае, если до данного вызова уже не были созданы подписки на событие ГИП и перенаправители события ГИП. Результатом последующих вызовов ***BioAPI_RedirectGUIEvents*** не будут дальнейшие вызовы ***BioSPI_SubscribeToGUIEvent,*** только если не были удалены все существующие перенаправители и подписки для ПБУ.

Если инфраструктура получает обратный вызов уведомления о событии выбора ГИП, то она должна сначала просмотреть все существующие перенаправители событий ГИП в поисках перенаправителя, где *GUISelectEventRedirected* имеет значение ВЕРНО и тот же дескриптор ПБУ, что и в событии ГИП. Применяется один из двух абзацев, указанных ниже.

Если совпадающих перенаправителей нет, то инфраструктура должна просмотреть все анонимные подписки на событие, созданные приложением, в поисках подписки с адресом обратного вызова, установленным на непустой указатель, для события выбора ГИП и либо дескриптором ПБУ, совпадающим с дескриптором ПБУ события, либо УУИД ПБУ, совпадающим с УУИД ПБУ события. Если совпадающая подписка есть,

то инфраструктура должна вызвать в данной подписке обработчик события выбора ГИП, в противном случае возвращать управление из ПБУ первоначальному обратному вызову с выходными параметрами по умолчанию.

Если совпадающий перенаправитель есть, то инфраструктура должна просмотреть все существующие именованные подписки на событие ГИП, созданные приложением, идентифицированным ИИР конечной точки подписчика в перенаправителе (пустой указатель — текущее приложение), в поисках подписки, в которой адрес обратного вызова для события выбора ГИП установлен на непустой указатель, УУИД подписки на событие выбора ГИП совпадает с УУИД подписки на событие выбора ГИП в перенаправителе, и УУИД ПБУ совпадает с УУИД ПБУ события. Если совпадающая подписка есть, то инфраструктура должна произвести обратный вызов обработчика события выбора ГИП, определенного в данной подписке, в противном случае возвращать управление из ПБУ первоначальному обратному вызову с выходными параметрами по умолчанию.

В обоих случаях после возвращения из обратного вызова приложения инфраструктура должна скопировать выходные параметры обратного вызова приложения в выходные параметры обратного вызова инфраструктуры с теми же наименованиями.

Случаи, перечисленные в пяти предыдущих абзацах, одинаково применимы как к событиям изменения состояния ГИП, так и к событиям выполнения ГИП.

Данная функция может быть вызвана только в том случае (для установленного УУИД ПБУ), если был произведен хотя бы один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого еще не был произведен соответствующий вызов ***BioAPI_BSPUnload.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

8.3.7.2 Параметры

SubscriberEndpointIRI (входной, дополнительный) — ИИР, идентифицирующий приложение, создавшее именованную подписку на событие ГИП. Параметр должен быть установлен на пустой указатель, если приложение-подписчик то же, что и текущее приложение.

GUIEventSubscriptionUuid (входной) — УУИД, идентифицирующий именованную подписку на событие ГИП. В некоторых случаях приложение получает данный УУИД из информации, возвращенной предшествующим вызовом ***BioAPI_QueryGUIEventSubscriptions.*** В других случаях приложение предоставляет известный УУИД.

BSPHandle (входной) — дескриптор присоединенной сессии ПБУ, связанный с событием ГИП. Данный вызов не оказывает влияния на события ГИП, относящиеся к другим присоединенным сессиям ПБУ.

GUISelectEventRedirected (входной) — флаг, определяющий то, находятся ли события выбора ГИП в области действий перенаправителя. Данный вызов оказывает влияние на события выбора ГИП тогда и только тогда, когда значение этого флага ВЕРНО.

GUIStateEventRedirected (входной) — флаг, определяющий то, находятся ли события изменения состояния ГИП в области действий перенаправителя. Данный вызов оказывает влияние на события выбора ГИП тогда и только тогда, когда значение этого флага ВЕРНО.

GUIProgressEventRedirected (входной) — флаг, определяющий то, находятся ли события выполнения ГИП в области действий перенаправителя. Данный вызов оказывает влияние на события выбора ГИП тогда и только тогда, когда значение этого флага ВЕРНО.

8.3.7.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.7.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.8 Функция BioAPI_SubscribeGUIEvents (БиоАПИ 2.1)

Данный пункт применяется только при использовании версии БиоАПИ 2.1.

```
BioAPI_RETURN BioAPI BioAPI_SubscribeToGUIEvents
  (const BioAPI_UUID *GUIEventSubscriptionUuid,
  const BioAPI_UUID *BSPUuid,
  const BioAPI_HANDLE *BSPHandle,
  BioAPI_GUI_SELECT_EVENT_HANDLER GUISelectEventHandler,
  const void *GUISelectEventHandlerCtx,
  BioAPI_GUI_STATE_EVENT_HANDLER GUIStateEventHandler,
  const void *GUIStateEventHandlerCtx,
  BioAPI_GUI_PROGRESS_EVENT_HANDLER GUIProgres Event-
HandLer,
  const void *GUIProgressEventHandlerCtx);
```

8.3.8.1 Описание

Данная функция создает «подписку на событие ГИП» для текущего приложения. Подписки на события ГИП поддерживаются инфраструктурой во время выполнения (обычно как записи внутренней таблицы) и дают возможность приложению получать уведомления о событиях ГИП по определенным установленным адресам обратных вызовов.

Приложение должно предоставить адреса обратных вызовов от одного до трех обработчиков событий ГИП (обработчик событий выбора ГИП, обработчик события изменения состояния ГИП, обработчик события выполнения ГИП), как и информацию, определяющую область действия подписки (либо загруженный ПБУ, либо присоединенная сессия ПБУ), таким образом, устанавливая предел на уведомления о событии ГИП, которые инфраструктура направит тем обработчикам событий ГИП. Все значения параметров, предоставленные в вызове данной функции, должны стать частью подписки на событие ГИП, поддерживаемого инфраструктурой.

В любом вызове данной функции должен быть предоставлен либо УУИД ПБУ, либо дескриптор ПБУ (но не оба). Если предоставлен дескриптор ПБУ, то подписка ограничена уведомлениями о событии ГИП, которые переносят этот дескриптор ПБУ. Если предоставлен УУИД ПБУ, то подписка ограничена уведомлениями о событии ГИП, которые переносят этот УУИД ПБУ, а также могут или не могут переносить дескриптор ПБУ. (Все уведомления о событии ГИП, которые переносят дескриптор ПБУ, также переносят УУИД ПБУ, но не наоборот).

Каждый вызов данной функции создает новую подписку на событие ГИП и не изменяет, а также не заменяет существующую подписку. Подписки на событие ГИП, определяющие дескриптор присоединенной сессии, должны быть автоматически удалены инфраструктурой, когда присоединенная сессия удалена. Подписки на события ГИП, определяющие УУИД ПБУ, должны быть автоматически удалены инфраструктурой, когда ПБУ выгружен. В любое время приложение может запросить инфраструктуру об удалении существующей подписки с помощью вызова ***BioAPI_UnredirectGUIEvents,*** предоставляя те же параметры, что были предоставлены в соответствующем вызове ***BioAPI_RedirectGUIEvents.***

Предоставленные в вызове контекстные адреса должны быть возвращены инфраструктурой приложению в последующих обратных вызовах обработчиков событий. В настоящем стандарте не рассматривается наделение смыслом конкретных адресов, но он определяет их для некоторых приложений.

Результатом многократного вызова данной функции, в любое время, не должны становиться многократные обработчики событий выбора ГИП, многократные обработчики событий изменения состояния ГИП или многократные обработчики событий выполнения ГИП, созданные для того же дескриптора ПБУ или для того же УУИД ПБУ. Сюда входят и случаи, когда одна подписка определяет УУИД ПБУ, а другая подписка определяет дескриптор присоединенной сессии того же ПБУ. Для любого входящего события ГИП это гарантирует определенность в том, вызов

какого обработчика события ГИП (если есть) должен производиться инфраструктурой.

Примечание – Предполагается недопустимость двух подписок с одинаковыми параметрами.

Если инфраструктура получает вызов данной функции, который успешно создал обработчик события ГИП, то она в ответ должна вызвать функцию ***BioSPI_SubscribeToGUIEvents*** ПБУ только в том случае, если до данного вызова уже не были созданы подписки на событие ГИП и перенаправители событий ГИП (см. 8.3.7). Результатом последующих вызовов ***BioAPI_SubscribeGUIEvents*** не будут дальнейшие вызовы ***BioSPI_SubscribeToGUIEvent,*** только если не были удалены все существующие подписки и перенаправители для данного ПБУ. ПБУ не нужно знать, сколько подписок на события ГИП требуется приложению, определяют ли эти подписки УУИД ПБУ или дескриптор ПБУ, не нужно знать типы, адреса обратных вызовов и контекстные адреса обработчиков событий ГИП, которые определены в данных подписках.

Данная функция может быть вызвана только в том случае (для установленного УУИД ПБУ), если был произведен хотя бы один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого еще не был произведен соответствующий вызов ***BioAPI_BSPUnload.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

Примечание – См. также С 7.

8.3.8.2 Параметры

GUIEventSubscriptionUuid (входной) — идентификатор подписки. Данный параметр должен быть установлен на пустой указатель (типичный случай) для анонимной подписки на событие ГИП (касающейся получения событий ГИП, созданных ПБУ и не перенаправленных). Данный параметр должен быть установлен на непустой указатель для названной подписки (касающейся получения уведомлений о перенаправленном событии ГИП и уведомления о событии ГИП, управляемом приложением).

BSPUuid (входной, дополнительный) — УУИД, определяющий ПБУ, которым ограничена подписка.

BSPHandle (входной, дополнительный) — дескриптор присоединенной сессии ПБУ, которой ограничена подписка. Данный параметр должен быть установлен на пустой указатель, если *GUIEventSubscriptionUuid* установлен на непустой указатель (названная подписка не может определять дескриптор присоединенной сессии).

GUISelectEventHandler (входной, дополнительный) — адрес обратного вызова функции приложения, которая должна получать от инфраструктуры уведомления о событии выбора ГИП.

GUISelectEventHandlerCtx (входной) — контекстный адрес, который должен возвращаться инфраструктурой приложению при каждом обратном вызове обработчика события выбора ГИП.

GUIStateEventHandler (входной, дополнительный) — адрес обратного вызова функции приложения, которая должна получать от инфраструктуры уведомления о событии изменения состояния ГИП.

GUIStateEventHandlerCtx (входной) — контекстный адрес, который должен возвращаться инфраструктурой приложению при каждом обратном вызове обработчика события изменения состояния ГИП.

GUIProgressEventHandler (входной, дополнительный) — адрес обратного вызова функции приложения, которая должна получать от инфраструктуры уведомления о событии выполнения ГИП.

GUIProgressEventHandlerCtx (входной) — контекстный адрес, который должен возвращаться инфраструктурой приложению при каждом обратном вызове обработчика события выполнения ГИП.

Должен быть определен только один из параметров *BSPUuid* и *BSPHandle*. Должен быть определен по крайней мере один из параметров *GUISelectEventHandler, GUIStateEventHandler* и *GUIProgressEventHandler*. Контекстный адрес должен быть определен только в том случае, если определен соответствующий адрес обратного вызова. Не определенные параметры должны быть установлены на пустой указатель.

8.3.8.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.8.4 Ошибки:

BioAPIERR_INVALID POINTER

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.9 Функция BioAPI_UnredirectGUIEvents (БиоАПИ 2.1)

Данный пункт применяется только при использовании версии БиоАПИ 2.1.

```
BioAPI_RETURN BioAPI BioAPI_UnredirectGUIEvents
  (const uint8_t *SubscriberEndpointIRI,
  const BioAPI_UUID *GUIEventSubscriptionUuid,
  BioAPI_HANDLE BSPHandle,
  BioAPI_BOOL GUISelectEventRedirected,
  BioAPI_BOOL GUIStateEventRedirected,
  BioAPI_BOOL GUIProgressEventRedirected);
```

8.3.9.1 Описание

Данная функция удаляет перенаправитель события ГИП, созданный посредством вызова ***BioAPI_RedirectGUIEvents.*** Определение именно этого удаленного перенаправителя в точности совпадает с параметрами вызова.

Примечание – Вызов данной функции не удалит перенаправитель, даже если существует лишь один перенаправитель, если значения параметров вызова полностью не совпадают со значениями параметров вызова ***BioAPI_RedirectGUIEvents,*** создавшего перенаправитель. Приложение должно помнить значения данных параметров.

Если инфраструктуре поступает вызов данной функции, который успешно удаляет перенаправитель события ГИП, то инфраструктура должна в ответ вызвать функцию ***BioSPI_UnsubscribeFromGUIEvents*** ПБУ только в том случае, если больше не существует подписок на события ГИП и перенаправителей событий ГИП, созданных для данного ПБУ.

Обычно у приложений нет необходимости вызывать данную функцию, потому что все перенаправители событий ГИП для указанной присоединенной сессии ПБУ автоматически удаляются инфраструктурой, когда удаляется присоединенная сессия. Данная функция может быть использована в том случае, если приложению необходимо модифицировать перенаправителей текущего события ГИП без удаления присоединенной сессии.

Данная функция может быть вызвана только в том случае (для установленного УУИД ПБУ), если был произведен хотя бы один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого еще не был произведен соответствующий вызов ***BioAPI_BSPUnload.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

Примечание – См. также С.7.

8.3.9.2 Параметры

Параметры данной функции имеют тот же смысл, что и параметры функции ***BioAPI_RedirectGUIEvents*** с теми же наименованиями, и используются инфраструктурой для идентификации существующего перенаправителя события ГИП, который должен быть удален.

8.3.9.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.9.4 Ошибки:

BioAPIERR_INVALID_POINTER

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.10 Функция BioAPI_UnsubscribeFromGUIEvents (БиоАПИ 2.1)

Данный пункт применяется только при использовании версии БиоАПИ 2.1.

```
BioAPI_RETURN BioAPI BioAPI_UnsubscribeFromGUIEvents
  (const BioAPI_UUID *GUIEventSubscriptionUuid,
  const BioAPI_UUID *BSPUuid,
  const BioAPI_HANDLE *BSPHandle,
  BioAPI_GUI_SELECT_EVENT_HANDLER GUISelectEventHandler,
  const void *GUISelectEventHandlerCtx,
  BioAPI_GUI_STATE_EVENT_HANDLER GUIStateEventHandler,
  const void *GUIStateEventHandlerCtx,
        BioAPI_GUI_PROGRESS_EVENT_HANDLER
    GUIProgressEventHandler
  const void *GUIProgressEventHandlerCtx);
```

8.3.10.1 Описание

Данная функция удаляет подписку на событие ГИП, созданную посредством вызова ***BioAPI_SubscribeToGUIEvents.*** Удаляется именно та подписка, определение которой точно соответствует параметрам вызова.

Примечание — Вызов данной функции не удалит подписку, даже если существует лишь одна подписка, если значения параметров вызова полностью не совпадают со значениями параметров вызова ***BioAPI_SubscribeToGUIEvents,*** создавшего подписку. Приложение должно помнить значения данных параметров.

Если инфраструктуре поступает вызов данной функции, который успешно удаляет подписку на событие ГИП, то инфраструктура должна в ответ вызвать функцию ***BioSPI_UnsubscribeFromGUIEvents*** ПБУ только в том случае, если больше не существует подписок на события ГИП и перенаправителей (см. 8.3.7) событий ГИП, созданных для данного ПБУ.

Обычно у приложений нет необходимости вызывать данную функцию, потому что все подписки на события ГИП для указанной присоединенной сессии ПБУ автоматически удаляются инфраструктурой, когда удаляется присоединенная сессия. Данная функция может быть использована в том случае, если приложению необходимо модифицировать подписку на текущее событие ГИП без удаления присоединенной сессии.

Пример – Данная функция может быть применена в следующих случаях: (1) приложение применяет управляемый приложением ГИП как при регистрации, так и при верификации/идентификации, но с разными обработчиками событий ГИП; (2) приложение использует ГИП, управляемый ПБУ, при регистрации, но управляемый приложением ГИП, при верификации/идентификации, или наоборот.

Данная функция может быть вызвана только в том случае (для установленного УУИД ПБУ), если был произведен хотя бы один вызов функции ***BioAPI_BSPLoad*** (для данного УУИД ПБУ), для которого еще не был произведен соответствующий вызов ***BioAPI_BSPUnload.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

Примечание – См. также С.7.

8.3.10.2 Параметры

Параметры данной функции имеют тот же смысл, что и параметры функции ***BioAPI_SubscribeGUIEvents*** с теми же наименованиями, и используются инфраструктурой для идентификации существующей подписки на событие ГИП, которая должна быть удалена.

8.3.10.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.10.4 Ошибки:

BioAPIERR_INVALID_POINTER

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

8.3.11 Функция BioAPI_EbableEventNotifications (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_EnableEventNotifications
  (const BioAPI_UUID *BSPUuid,
  BioAPI_EVENT_MASK Events);
```

8.3.11.1 Описание

Функция разрешает события, определенные маской событий и поступающие от ПБУ, идентифицированного УУИД ПБУ, и деактивирует все остальные события, поступающие от этого же ПБУ.

События разрешены или деактивированы только для приложения, вызывающего данную функцию. Если существуют другие приложения, использующие инфраструктуру БиоАПИ или указанный ПБУ одновременно, то на такие приложения вызов функции ***BioAPI_EnableEvent-Notification*** влияния не окажет.

Данная функция может быть вызвана в любое время после ***BioAPI_Init*** или ***BioAPI_InitEndpoint,*** даже до загрузки указанного ПБУ. Маска событий, созданная вызовом данной функции, остается действующей до тех пор, пока эта функция не будет снова вызвана для того же ПБУ.

Данная функция должна вызываться только в том случае, если был произведен хотя бы один вызов ***BioAPI_Init*** или ***BioAPI_InitEndpoint,*** для которого еще не был произведен соответствующий вызов ***BioAPI_ Terminate.***

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ни одному ПБУ.

8.3.11.2 Параметры

BSPUuid (входной) — УУИД ПБУ.

Events (входной) — маска, обозначающая разрешенные события.

8.3.11.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

8.3.11.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11».

Подпункт 9.2.1.1 Первый — четвертый абзацы (определения параметров) изложить в новой редакции:

«*BSPUuid (входной)* — УУИД ПБУ, вызывающего событие.

UnitID (входной) — ИД модуля БиоАПИ, связанного с данным событием.

UnitSchema (входной) — указатель на модуль схемы модуля БиоАПИ, связанного с данным событием.

EventType (входной) — произошедшее событие типа BioAPI_EVENT».

Подраздел 9.2 дополнить пунктами — 9.2.4—9.2.6.5:

«9.2.4 BioSPI_GUI_PROGRESS_EVENT_HANDLER (БиоАПИ 2.1)

Данный пункт применяется только при использовании номера версии БиоАПИ 2.1.

9.2.4.1 Функция обратного вызова

```
typedef BioAPI_RETURN (BioAPI
*BioSPI_GUI_PROGRESS_EVENT_HANDLER)
  (const BioAPI_UUID *BSPUuid,
  BioAPI_UNIT_ID UnitID,
  const BioAPI_HANDLE *BSPHandle,
  BioAPI_GUI_OPERATION Operation,
  BioAPI_GUI_SUBOPERATION Suboperation,
  BioAPI_BIR_PURPOSE Purpose,
```

BioAPI_GUI_MOMENT Moment,
uint8_t SuboperationProgress,
const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
const uint8_t *Text,
BioAPI_GUI_RESPONSE *Response);

9.2.4.2 Описание

Данный пункт описывает тип указателя на функцию для функции обработчика события ГИП инфраструктуры, который обрабатывает обратные вызовы уведомления о событии выполнения ГИП, поступающие от ПБУ. Для того чтобы получать уведомления о событии выполнения ГИП, инфраструктура должна зарегистрировать функцию обратного вызова типа BioSPI_GUI_PROGRESS_EVENT_HANDLER путем предоставления адреса обратного вызова функции в вызове ***BioSPI_SubscribeToGUIEvents*** (см. 9.3.3.3).

ПБУ производит обратный вызов функции инфраструктуры этого типа для того, чтобы уведомить инфраструктуру (в конечном счете, приложение) о том, что создано событие выполнения ГИП в процессе выполнения функции ПБУ.

ПБУ может создавать события выполнения ГИП только в процессе выполнения вызова ***BioSPI_Capture, BioSPI_Process, BioSPI_Cre-ateTemplate, BioSPI_VerifyMatch, BioSPI_IdentifyMatch, BioSPI_Verify, BioSPI_Identify*** или ***BioSPI_Enroll.*** Они могут быть созданы в любое время, даже многократно, в процессе любой подоперации.

9.2.4.3 Параметры

Параметры данного типа указателя на функцию совпадают с параметрами BioAPI_GUI_PROGRESS_EVENT_HANDLER, кроме того, что отсутствует параметр контекстного адреса.

Примечание – См. также раздел 7.

9.2.4.4 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

9.2.4.5 Ошибки

BioAPIERR_USER_CANCELLED
BioAPIERR_FUNCTION_FAILED

9.2.5 BioSPI_GUI_SELECT_EVENT_HANDLER (БиоАПИ 2.1)

Данный пункт применяется только при использовании номера версии БиоАПИ 2.1.

9.2.5.1 Функция обратного вызова

```
typedef BioAPI_RETURN (BioAPI *BioSPI_GUI_SELECT_EVENT_
HANDLER)
  (const BioAPI_UUID *BSPUuid,
  BioAPI_UNIT_ID UnitID,
  const BioAPI_HANDLE *BSPHandle,
  BioAPI_GUI_ENROLL_TYPE EnrollType,
  BioAPI_GUI_OPERATION Operation,
  BioAPI_GUI_MOMENT Moment,
  BioAPI_RETURN ResultCode,
  int32_t MaxNumEnrollSamples,
  BioAPI_BIR_SUBTYPE_MASK SelectableInstances,
  BioAPI_BIR_SUBTYPE_MASK *SelectedInstances,
  BioAPI_BIR_SUBTYPE_MASK CapturedInstances,
  const uint8_t *Text,
  BioAPI_GUI_RESPONSE *Response);
```

9.2.5.2 Описание

Данный пункт описывает тип указателя на функцию для функции обработчика события ГИП инфраструктуры, который обрабатывает обратные вызовы уведомления о событии выбора ГИП, поступающие от ПБУ. Для того чтобы получать уведомления о событии выбора ГИП, инфраструктура должна зарегистрировать функцию обратного вызова типа BioSPI_GUI_SELECT_EVENT_HANDLER путем предоставления адреса обратного вызова функции в вызове ***BioSPI_SubscribeToGUIEvents*** (см. 9.3.3.3).

ПБУ производит обратный вызов функции инфраструктуры этого типа для того, чтобы уведомить инфраструктуру (в конечном счете, приложение) о том, что создано событие выбора ГИП в процессе выполнения функции ПБУ.

ПБУ может создавать события выбора ГИП только в процессе выполнения вызова ***BioSPI_Capture, BioSPI_Verify, BioSPI_Identify*** или ***BioSPI_Enroll.*** Событие выбора ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START генерируется перед запуском каждого цикла подопераций, а событие выбора ГИП со значением момента BioAPI_GUI_MOMENT_AFTER_END генерируется после окончания каждого цикла подопераций (см. подраздел 7.66).

9.2.5.3 Параметры

Параметры данного типа указателя на функцию совпадают с параметрами BioAPI_GUI_SELECT_EVENT_HANDLER, кроме того, что отсутствует параметр контекстного адреса.

Примечание — См. также раздел 7.

9.2.5.4 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

9.2.5.5 Ошибки

BioAPIERR_USER_CANCELLED

BioAPIERR_FUNCTION_FAILED

9.2.6 BioSPI_GUI_STATE_EVENT_HANDLER (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

9.2.6.1 Функция обратного вызова

```
typedef BioAPI_RETURN (BioAPI
*BioSPI_GUI_STATE_EVENT_HANDLER)
   (const BioAPI_UUID *BSPUuid,
   BioAPI_UNIT_ID UnitID,
   const BioAPI_HANDLE *BSPHandle,
   BioAPI_GUI_OPERATION Operation,
   BioAPI_GUI_SUBOPERATION Suboperation,
   BioAPI_BIR_PURPOSE Purpose,
   BioAPI_GUI_MOMENT Moment,
   BioAPI_RETURN ResultCode,
   int32_t EnrollSampleIndex,
   const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
   const uint8_t *Text,
   BioAPI_GUI_RESPONSE *Response,
   int32_t *EnrollSampleIndexToRecapture);
```

9.2.6.2 Описание

Данный пункт описывает тип указателя на функцию для функции обработчика события ГИП инфраструктуры, который обрабатывает обратные вызовы уведомления о событии изменения состояния ГИП, поступающие от ПБУ. Для того чтобы получать уведомления о событии изменения состояния ГИП, инфраструктура должна зарегистрировать функцию обратного вызова типа BioSPI_GUI_STATE_ EVENT_ HANDLER путем предоставления адреса обратного вызова функции при вызове ***BioSPI_SubscribeToGUIEvents*** (см. 9.3.3.3).

ПБУ производит обратный вызов функции инфраструктуры этого типа для того, чтобы уведомить инфраструктуру (и, в конечном счете, приложение) о том, что создано событие изменения состояния ГИП в процессе выполнения функции ПБУ.

ПБУ может создавать события изменения состояния ГИП только в процессе выполнения вызова ***BioSPI_Capture, BioSPI_Verify, BioSPI_Identify*** или ***BioSPI_Enroll.*** Событие изменения состояния ГИП со значением момента BioAPI_GUI_MOMENT_BEFORE_START генерируется перед запуском каждой подоперации, а событие выбора ГИП со значением момента BioAPI_GUI_MOMENT_AFTER_END генерируется после окончания каждой подоперации (см. подраздел 7.66).

9.2.6.3 Параметры

Параметры указателя на функцию данного типа совпадают с параметрами BioAPI_GUI_ STATE_EVENT_HANDLER, за исключением отсутствия параметра контекстного адреса.

Примечание – См. также раздел 7.

9.2.6.4 Возвращаемое значение

Значение BioAPI_RETURN указывает на то, что функция была выполнена успешно, или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

9.2.6.5 Ошибки

BioAPIERR_USER_CANCELLED

BioAPIERR_FUNCTION_FAILED».

Подпункт 9.3.1.1.1 дополнить абзацем (после первого):

«При использовании БиоАПИ версии 2.1 ПБУ, получающий вызов данной функции, должен для каждого существующего модуля БиоАПИ немедленно выслать одно уведомление о событии «установка» путем обратного вызова функции, предоставленной инфраструктурой, в адрес, который указывается в ***BioAPINotifyCallback.*** Если биометрическое приложение предоставило в вызове функции ***BioAPI_BSPLoad*** обработчик события, то инфраструктура, в свою очередь, произведет обратный вызов обработчика событий приложения. Если аппаратный компонент для данного модуля БиоАПИ не предоставлен, то событие «установки» не должно вызываться до тех пор, пока оно не будет подключено».

Подпункт 9.3.1.3.2 дополнить абзацем (перед последним примечанием):

«При использовании БиоАПИ версии 2.1 параметр не может иметь значение BioAPI_INVALID_BSP_HANDLE».

Пункт 9.3.1 дополнить подпунктами — 9.3.1.8, 9.3.1.9:

«9.3.1.8 BioSPI_Control (БиоАПИ 2.1)

Данный подпункт применяется только при использовании БиоАПИ версии 2.1.

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

Примечание — Подробности определения данной функции представлены в 8.1.13, BioAPI_Control.

9.3.1.9 BioSPI_Transform (БиоАПИ 2.1)

Данный подпункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioSPI_Transform
  (BioAPI_HANDLE BSPHandle,
  const BioAPI_UUID *OperationUUID,
  const BioAPI_INPUT_BIR *InputBIRs,
  uint32_t NumberOfInputBIRs,
  BioAPI_BIR_HANDLE **OutputBIRs,
  uint32_t *NumberOfOutputBIRs);
```

Примечание — Подробности определения данной функции представлены в 8.1.14, BioAPI_Transform».

Подпункт 9.3.3.2. Заголовок изложить в новой редакции:

«9.3.3.2 BioSPI_ SetGUICallbacks (БиоАПИ 2.0)»;

дополнить абзацем (перед первым):

«Данный подпункт применяется только при использовании БиоАПИ версии 2.0».

Пункт 9.3.3 дополнить подпунктами — 9.3.3.3—9.3.3.4.1:

«9.3.3.3 BioSPI_SubscribeToGUIEvents (БиоАПИ 2.1)

Данный подпункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioSPI_SubscribeToGUIEvents
  (const BioAPI_UUID *BSPUuid,
  BioSPI_GUI_SELECT_EVENT_HANDLER FwGUISelectEventHandler,
  BioSPI_GUI_STATE_EVENT_HANDLER FwGUIStateEventHandler,
  BioSPI_GUI_PROGRESS_EVENT_HANDLER FwGUIProgress-
EventHandler);
```

Данная функция предоставляет ПБУ адреса обратных вызовов обработчика событий выбора ГИП, обработчика событий изменения состояния ГИП и обработчика событий выполнения ГИП. Все три адреса не должны быть установлены на пустой указатель.

После вызова данной функции и до последующего вызова ***BioSPI_UnsubscribeFromGUIEvents*** или ***BioSPI_BSPUnload*** для каждого события ГИП, которое генерирует ПБУ в процессе выполнения вызовов функции ПБУ, ПБУ должен уведомить об этом событии инфраструктуру путем ответного вызова обработчика событий ГИП инфраструктуры, соответствующего типу события ГИП. Кроме того, любой ГИП, управляемый ПБУ, должен быть деактивирован в течение этого времени.

Параметр *BSPUuid* предоставляется только в качестве показателя надежности. ПБУ должен подтвердить, что значение данного параметра соответствует значению УУИД продукта ПБУ.

Эта функция, в отличие от функции ***BioAPI_SubscribeToGUIEvents,*** не должна создавать новую подписку на каждый вызов. На каждый вызов ПБУ должен просто заменить адреса старых обратных вызовов (изначально пустой указатель) адресами, предоставленными в вызове.

Единственный вызов ***BioAPI_UnsubscribeFromGUIEvents*** очистит все три адреса обратных вызовов. Необходим только один такой вызов, даже в том случае, если функция ***BioAPI_SubscribeToGUIEvents*** вызывалась многократно.

Примечание — После того как инфраструктура вызвала данную функцию, она уже не вызовет ее до тех пор, пока не будет произведен вызов ***BioSPI_UnsubscribeFromGUIEvents*** или ***BioSPI_BSPUnload.*** Инфраструктура передает все три адреса обратного вызова при каждом вызове данной функции.

9.3.3.3.1 Параметры

BSPUuid (входной) — УУИД ПБУ.

FwGUISelectEventHandler (входной) — адрес обратного вызова функции инфраструктуры, которая получает уведомления о событии выбора ГИП от ПБУ.

FwGUIStateEventHandler (входной) — адрес обратного вызова функции инфраструктуры, которая получает уведомления о событии изменения состояния ГИП от ПБУ.

FwGUIProgressEventHandler (входной, необязательный) — адрес обратного вызова функции инфраструктуры, которая получает уведомления о событии выполнения ГИП от ПБУ.

9.3.3.4 BioSPI_UnsubscribeFromGUIEvents (БиоАПИ 2.1)

Данный подпункт применяется только при использовании БиоАПИ версии 2.1.

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

Данная функция очищает адреса обратных вызовов обработчика событий выбора ГИП, обработчика событий изменения состояния ГИП и обработчика событий выполнения ГИП.

После вызова данной функции ПБУ должен прекратить уведомлять инфраструктуру о событиях ГИП.

Параметр *BSPUuid* предоставляется только в качестве показателя надежности. ПБУ должен подтвердить, что значение данного параметра соответствует значению УУИД продукта ПБУ.

9.3.3.4.1 Параметры

BSPUuid (входной) — УУИД ПБУ.

Примечание – См. также раздел 7».

Пункт 10.1.2. Таблицу 3 для поля BSPUUID изложить в новой редакции:

Имя поля	Тип данных поля	Описание
BSPUUID	UINT8_T (16)	УУИД, однозначно идентифицирующий ПБУ. Данное поле должно присутствовать только при использовании БиоАПИ версии 2.0
BSPProductUUID	UINT8_T (16)	УУИД, однозначно идентифицирующий ПБУ как продукт программного обеспечения. Данное поле должно присутствовать только при использовании БиоАПИ версии 2.1

таблицу 3 дополнить элементами схемы ПБУ (перед сноской):

Имя поля	Тип данных поля	Описание
MaxNumEnrollInstances	UINT32_T	Максимальное число различных экземпляров, для которых ПБУ может создать контрольные шаблоны за одну операцию регистрации. Данное поле должно присутствовать только при использовании БиоАПИ версии 2.1

Окончание

Имя поля	Тип данных поля	Описание
HostingEndpointIRI	UINT8_T	ИИР, идентифицирующий инфраструктуру, чей регистр компонентов содержит регистрацию ПБУ. Данное поле должно присутствовать только при использовании БиоАПИ версии 2.1
BSPAccessUUID	BioAPI_UUID	УУИД, уникальный в области применения приложения, который приложение может использовать для обращения к ПБУ вместо УУИД продукта ПБУ. Данное поле должно присутствовать только при использовании БиоАПИ версии 2.1

Подраздел 10.2 дополнить пунктами — 10.2.3—10.2.7:

«10.2.3 Функция BioAPI_RegisterBSP (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_RegisterBSP
  (const uint8_t *HostingEndpointIRI,
  const BioAPI_BSP_SCHEMA *BSPSchema, BioAPI_BOOL Update);
```

10.2.3.1 Описание

В спецификации, предоставленной настоящим стандартом, данная функция выполняет те же действия, что и функция ***BioAPI_Util_InstallBSP*** с параметром *Action*, установленным на BioAPI_INSTALL_ACTION_INSTALL или BioAPI_INSTALL_ACTION_REFRESH. Значение дополнительного параметра *HostingEndpointIRI* должно быть проигнорировано инфраструктурами, соответствующими этому разделу настоящего стандарта, и установлено на пустой указатель приложением. Данный параметр предоставлен для поддержки стандартов межсетевого обмена.

В отличие от функции ***BioAPI_Util_InstallBSP*** данная функция не содержит параметр *Error*. Информация об ошибке предоставлена в возвращаемом значении.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБУ.

10.2.3.2 Параметры

HostingEndpointEndpointIRI — данный параметр должен быть проигнорирован инфраструктурами, соответствующими этому разделу настоящего стандарта, и установлен на пустой указатель приложением. Данный параметр предоставлен для поддержки стандартов межсетевого обмена.

BSPSchema (входной) — указатель на элементы схемы ПБУ, определенные в 7.16, описывающие свойства и характеристики устанавливаемого ПБУ.

Update (входной) — булево значение, указывающее на то, должна ли функция обновить существующую регистрацию схемы ПБУ или должна создать новую регистрацию. Если значение данного параметра равно нулю и схема ПБУ находится в реестре компонентов, то функция должна возвратить значение BioAPIERR_COMPONENT_ALREADY_REGISTERED. Если значение данного параметра не равно нулю и схема ПБУ не находится в реестре компонентов, то функция должна возвратить значение BioAPIERR_COMPONENT_NOT_REGISTERED.

10.2.3.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

10.2.3.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

10.2.4 Функция BioAPI_UnregisterBSP (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_UnregisterBSP
  (const uint8_t *HostingEndpointIRI,
  const BioAPI_UUID *BSPUuid);
```

10.2.4.1 Описание

В спецификации, предоставленной настоящим стандартом, данная функция выполняет те же действия, что и функция ***BioAPI_Util_InstallBSP*** с параметром *Action*, установленным на BioAPI_INSTALL_ACTION_UNINSTALL. Значение дополнительного параметра *HostingEndpointIRI* должно быть проигнорировано инфраструктурами, соответствующими этому разделу настоящего стандарта, и установлено на пустой указатель приложением. Данный параметр предоставлен для поддержки стандартов межсетевого обмена.

В отличие от функции ***BioAPI_Util_UninstallBSP*** данная функция не содержит параметр *Error*. Информация об ошибке предоставлена в возвращаемом значении.

Приложения, соответствующие настоящему стандарту, должны либо не вызывать функцию, либо (если вызывают ее) должны установить данный параметр *HostingEndpointIRI* на пустой указатель. Функция предоставлена для поддержки стандартов межсетевого обмена.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБУ.

10.2.4.2 Параметры

HostingEndpointEndpointIRI — данный параметр должен быть проигнорирован инфраструктурами, соответствующими этому разделу настоящего стандарта, и установлен на пустой указатель приложением. Данный параметр предоставлен для поддержки стандартов межсетевого обмена.

BSPSchema (входной) — указатель на УУИД ПБУ, чья регистрация схемы ПБУ должна быть удалена из реестра компонентов.

10.2.4.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

10.2.4.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

10.2.5 Функция BioAPI_RegisterBFP (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_RegisterBFP
  (const uint8_t *HostingEndpointIRI,
  const BioAPI_BFP_SCHEMA *BFPSchema,
  BioAPI_BOOL Update);
```

10.2.5.1 Описание

Данная функция выполняет те же действия, что и функция ***BioAPI_Util_InstallBFP*** с параметром *Action*, установленным на BioAPI_INSTALL_ACTION_INSTALL или BioAPI_INSTALL_ACTION_REFRESH. Значение дополнительного параметра *HostingEndrpointIRI* должно быть проигнорировано инфраструктурами, соответствующими этому разделу настоящего стандарта, и установлено на пустой указатель приложением. Данный параметр предоставлен для поддержки стандартов межсетевого обмена.

В отличие от функции ***BioAPI_Util_UninstallBFP*** данная функция не содержит параметр *Error*. Информация об ошибке предоставлена в возвращаемом значении.

Приложения, соответствующие настоящему стандарту, должны либо не вызывать функцию, либо (если вызывают ее) должны установить данный параметр *HostingEndpointIRI* на пустой указатель. Функция предоставлена для поддержки стандартов межсетевого обмена.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБФ.

10.2.5.2 Параметры

HostingEndpointEndpointIRI — данный параметр должен быть проигнорирован инфраструктурами, соответствующими этому разделу настоящего стандарта, и установлен на пустой указатель приложением. Данный параметр предоставлен для поддержки стандартов межсетевого обмена.

BFPSchema (входной) — указатель на элементы схемы ПБФ, определенные в 7.16, описывающие свойства и характеристики устанавливаемого ПБФ.

Update (входной) — булево значение, определяющее должна ли функция обновить существующую регистрацию схемы ПБФ или должна создать новую регистрацию. Если значение данного параметра равно нулю и схема ПБФ находится в реестре компонентов, то функция должна возвратить значение BioAPIERR_COMPONENT_ALREADY_REGISTERED. Если значение данного параметра не равно нулю и схема ПБФ не находится в реестре компонентов, то функция должна возвратить значение BioAPIERR_COMPONENT_NOT_REGISTERED.

10.2.5.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

10.2.5.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

10.2.6 Функция BioAPI_UnregisterBFP (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_UnregisterBFP
  (const uint8_t *HostingEndpointIRI,
  const BioAPI_UUID *BFPUuid);
```

10.2.6.1 Описание

В спецификации, предоставленной настоящим стандартом, данная функция выполняет те же действия, что и функция ***BioAPI_Util_InstallBFP***

с параметром *Action*, установленным на BioAPI_INSTALL_ACTION_ UNINSTALL. Значение дополнительного параметра *HostingEndpointIRI* должно быть проигнорировано инфраструктурами, соответствующими этому разделу настоящего стандарта, и установлено на пустой указатель приложением. Данный параметр предоставлен для поддержки стандартов межсетевого обмена.

В отличие от функции ***BioAPI_Util_UninstallBFP*** данная функция не содержит параметр *Error*. Информация об ошибке предоставлена в возвращаемом значении.

Приложения, соответствующие настоящему стандарту, должны либо не вызывать функцию, либо (если вызывают ее) должны установить данный параметр *HostingEndpointIRI* на пустой указатель. Функция предоставлена для поддержки стандартов межсетевого обмена.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБФ.

10.2.6.2 Параметры

HostingEndpointEndpointIRI — данный параметр должен быть проигнорирован инфраструктурами, соответствующими этому разделу настоящего стандарта, и установлен на пустой указатель приложением. Данный параметр предоставлен для поддержки стандартов межсетевого обмена.

BFPSchema (входной) — указатель на УУИД ПБФ, чья регистрация схемы ПБФ должна быть удалена из реестра компонентов.

10.2.6.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

10.2.6.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11.

10.2.7 Функция BioAPI_GetLastErrorInfo (БиоАПИ 2.1)

Данный пункт применяется только при использовании БиоАПИ версии 2.1.

```
BioAPI_RETURN BioAPI BioAPI_GetLastErrorInfo
  (BioAPI_ERROR_INFO *ErrorInfo);
```

10.2.7.1 Описание

Данная функция записывает в элементы выходного параметра *ErrorInfo* (см. 7.72) информацию о последнем типе ошибки, инициировавшем функцию БиоАПИ возвратить код ошибки.

Данная функция в основном предназначена для предоставления приложению диагностической информации. Так как возможные значения

элементов *ErrorInfo* не стандартизированы и могут меняться в зависимости от инфраструктуры и реализации, нормальные приложения БиоАПИ не должны основываться на этих значениях при принятии каких-либо важных решений по обработке данных.

Данная функция обрабатывается в инфраструктуре БиоАПИ и не передается ПБУ.

10.2.7.2 Параметры

ErrorInfo (выходной) — структура данных типа BioAPI_ERROR_INFO, содержащая информацию о последней произошедшей ошибке.

10.2.7.3 Возвращаемое значение

Значение BioAPI_RETURN указывает на успешное выполнение функции или определяет тип ошибки. Значение BioAPI_OK указывает на успешное выполнение функции. Все остальные значения описывают тип ошибки.

10.2.7.4 Ошибки

Данные об обработке ошибок БиоАПИ приведены в разделе 11».

Пункт 11.2.3 дополнить абзацами:

«#define BioAPIERR_IDENTIFY_IN_PROGRESS (0x000120)

Идентификация находится в процессе выполнения. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_LOW_QUALITY_REFERENCE_TEMPLATE (0x000121)

Качество контрольного образца низкое. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_NO_GUI_EVENT_HANDLER (0x000122)

Обработчик событий ГИП не установлен. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_TRANSFORMATION_NOT_SUPPORTED (0x000121)

ПБУ не поддерживает трансформацию ЗБИ. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_INVALID_ATTACH_SESSION (0x000122)

Присоединенная сессия недействительна. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_COMPONENT_ALREADY_REGISTERED (0x000123)

Схема ПБУ или ПБФ уже присутствует в реестре компонентов. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_COMPONENT_NOT_REGISTERED (0x000124)

Схема ПБУ или ПБФ не присутствует в реестре компонентов. Данное определение применяется только при использовании БиоАПИ версии 2.1».

Пункт 11.2.7 дополнить абзацами (после примечания):

«#define BioAPIERR_TOO_EARLY (0x000502)

Действия пользователя произошли слишком рано. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_TOO_LATE (0x000503)

Действия пользователя произошли слишком поздно. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_TOO_FAST (0x000504)

Действия пользователя произошли слишком быстро. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_TOO_SLOW (0x000505)

Действия пользователя произошли слишком медленно. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_TOO_LONG (0x000506)

Объем полученных данных слишком велик. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_TOO_SHORT (0x000507)

Объем полученных данных слишком мал. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_TOO_LARGE (0x000508)

Объем полученных данных слишком велик. Данное определение применяется только при использовании БиоАПИ версии 2.1.

#define BioAPIERR_TOO_SMALL (0x000509)

Объем полученных данных слишком мал. Данное определение применяется только при использовании БиоАПИ версии 2.1».

Приложение А. Пункт А.1.1 изложить в новой редакции:

«А.1.1 Соответствие требованиям настоящего стандарта состоит из следующих пяти классов:

- БиоАПИ совместимое биометрическое приложение;
- БиоАПИ 2.0 совместимая инфраструктура БиоАПИ;
- БиоАПИ 2.1 совместимая инфраструктура БиоАПИ;
- БиоАПИ 2.0 совместимый ПБУ, включающий в себя один из следующих подклассов:
  - БиоАПИ 2.0 совместимый ПБУ для верификации;
  - БиоАПИ 2.0 совместимый ПБУ для идентификации;
  - БиоАПИ 2.0 совместимый ПБУ для получения данных;

- БиоАПИ 2.0 совместимый механизм верификации;
- БиоАПИ 2.0 совместимый механизм идентификации.

- БиоАПИ 2.1 совместимый ПБУ, включающий в себя один из следующих подклассов:
- БиоАПИ 2.1 совместимый ПБУ для верификации;
- БиоАПИ 2.1 совместимый ПБУ для идентификации;
- БиоАПИ 2.1 совместимый ПБУ для получения данных;
- БиоАПИ 2.1 совместимый механизм верификации;
- БиоАПИ 2.1 совместимый механизм идентификации».

Пункт А.3.2 после слов «Для соответствия БиоАПИ» дополнить словами: «(или версия 2.0 или версия 2.1)».

Пункт А.3.3 изложить в новой редакции:

«А.3.3 Соответствующая инфраструктура БиоАПИ 2.0 должна поддерживать все обязательные и дополнительные опции, установленные настоящим стандартом, кроме подпунктов и определений, которые указаны как применимые только при использовании версии 2.1».

Подраздел А.3 дополнить пунктами — А.3.4, А.3.5:

«А.3.4 Соответствующая инфраструктура БиоАПИ 2.1 должна поддерживать все обязательные и дополнительные опции, установленные настоящим стандартом, кроме подпунктов и определений, которые указаны как применимые только при использовании версии 2.0.

А.3.5 Не определено никаких подгрупп функций, соответствующих инфраструктуре».

Подраздел А.4. Первый абзац. Заменить слова: «Для соответствия настоящему стандарту» на «Для соответствия БиоАПИ (или версия 2.0, или версия 2.1)»;

дополнить абзацами (после первого):

«ПБУ, совместимые с БиоАПИ 2.0 (любого совместимого подкласса), должны поддерживать все обязательные функции, установленные настоящим стандартом для соответствующего подкласса, кроме подпунктов и определений, которые указаны как применимые только при использовании версии 2.1.

ПБУ, совместимые с БиоАПИ 2.1 (любого совместимого подкласса), должны поддерживать все обязательные функции, установленные настоящим стандартом для соответствующего подкласса, кроме подпунктов и определений, которые указаны как применимые только при использовании версии 2.0»;

таблицу А.1 для области «Функции управления обратными вызовами и событиями» и функции ***«BioSPI_SetGUICallbacks»*** изложить в новой редакции и дополнить следующими функциями:

Функция	ПБУ для			Механизм верификации	Механизм идентификации
	верификации	идентификации	получения данных		
BioSPI_SetGUICallbacks (БиоАПИ 2.0)					
BioSPI_SubscribeToGUIEvents (БиоАПИ 2.1)					
BioSPI_UnsubscribeFromGUIEvents (БиоАПИ 2.1)					

таблицу А.1 дополнить функциями — «События ГИП» (в конце таблицы):

Функция	ПБУ для			Механизм верификации	Механизм идентификации
	верификации	идентификации	получения данных		
События ГИП					
События выбора ГИП (БиоАПИ 2.1)					
События изменения состояния ГИП (БиоАПИ 2.1)					
События выполнения ГИП (БиоАПИ 2.1)					

Подпункт А.4.6.2. Таблица А.2. Подфункцию «Обратные вызовы потокового ГИП» изложить в новой редакции и дополнить следующими подфункциями:

Возможность	Поддерживается	Не поддерживается
Обратные вызовы потокового ГИП (БиоАПИ 2.1)		

Окончание

Возможность	Поддерживается	Не поддерживается
Генерирование событий выполнения ГИП в течение подопераций захвата (БиоАПИ 2.1)		
Генерирование событий выполнения ГИП в течение подопераций идентификации сопоставления (БиоАПИ 2.1)		

Приложение В. Пункт В.5. Заменить слова: «{iso registration-authority cbeff(19785) organization(0) 257 bir(1) patron-format (6)} на «{iso registration-authority cbeff(19785) biometric-organization(0) jtc1-sc37(257) patron-format(1) bioAPI(8)}».

Пункт В.10. Таблицу В.1 для наименований «BioAPI_BIR_BIOMETRIC_TYPE (БиоАПИ, 7.8)» и «BioAPI_BIR_SUBTYPE (БиоАПИ, 7.14)» изложить в новой редакции:

Наименование поля БиоАПИ, ссылки БиоАПИ и поля формата постоянного клиента	Наименование элемента данных ЕСФОБД	Длина, байт	Абстрактное значение	Кодировка*	Ссылка ЕСФОБД
BioAPI_BIR_BIOMETRIC_TYPE (БиоАПИ, 7.8)	CBEFF_BDB_biometric_type	4	NO BIOTYPE AVAILABLE (доступных биотипов нет)	'00 00 00 00'	
			MULTIPLE_BIOMETRIC_TYPES	'00 00 00 01'	
			FACE (изображение лица)	'00 00 00 02'	
			VOICE (голос)	'00 00 00 04'	
			FINGER (изображение отпечатка пальца)	'00 00 00 08'	
			IRIS (изображение радужной оболочки глаза)	'00 00 00 10'	
			RETINA (сетчатка)	'00 00 00 20'	
			HAND GEOMETRY (геометрия контура кисти руки)	'00 00 00 40'	

Окончание

Наименование поля БиоАПИ, ссылки БиоАПИ и поля формата постоянного клиента	Наименование элемента данных ЕСФОБД	Длина, байт	Абстрактное значение	Кодировка	Ссылки ЕСФОБД
BioAPI_BIR_BIOMETRIC_TYPE (БиоАПИ, 7.8)	CBEFF_BDB_biometric_type	4	SIGNATURE SIGN (динамика подписи)	'00 00 00 80'	
			KEYSTROKE (нажатие клавиши)	'00 00 01 00'	
			LIP MOVEMENT (движения губ)	'00 00 02 00'	
			RESERVED (зарезервировано)	'00 00 04 00'	
			RESERVED (зарезервировано)	'00 00 08 00'	
			GAIT (походка)	'00 00 10 00'	
			VEIN (вена)	'00 00 20 00'	
			DNA (ДНК)	'00 00 40 00'	
			EAR (ухо)	'00 00 80 00'	
			FOOT (ступня)	'00 01 00 00'	
			SCENT (запах)	'00 02 00 00'	
			OTHER (другое)	'40 00 00 00'	
			PASSWORD (пароль)	'80 00 00 00'	
BioAPI_BIR_SUBTYPE (БиоАПИ, 7.14)	CBEFF_BDB_biometric_subtype	1	NO_SUBTYPE_AVAILABLE	'00'	
			VEIN_ONLY_MASK	'80'	
			LEFT_MASK	'01'	
			RIGHT_MASK	'02'	
			THUMB (большой палец)	'04'	
			POINTERFINGER (указательный палец)	'08'	
			MIDDLEFINGER (средний палец)	'10'	
			RINGFINGER (безымянный палец)	'20'	
			LITTLEFINGER (мизинец)	'40'	
			VEIN_PALM (ладонь)	'04'	
			VEIN_BACKOFHAND (тыльная сторона ладони)	'08'	
			VEIN_WRIST (запястье)	'10'	

Приложение С. Подраздел С.7 изложить в новой редакции:

«С.7 Анализ интерфейса пользователя (БиоАПИ 2.1)

Анализ и примеры в данном подразделе применяются только при использовании БиоАПИ версии 2.1.

С.7.1 Общие положения

Интерфейс пользователя для паролей и личных идентификационных номеров является достаточно простым, но для биометрической технологии он может быть весьма сложным и в значительной степени зависеть от применяемой технологии, требуя многочисленные взаимодействия с пользователем. Некоторые биометрические технологии демонстрируют поток данных пользователю (например, лицо и голос), а другие требуют от пользователя подтверждения каждого взятого образца (например, лицо, голос, подпись). В процессе регистрации некоторые технологии верифицируют каждый взятый образец на основе предыдущих образцов (тест-верификация), используя определенный алгоритм сопоставления. Число образцов, взятых для конкретного назначения (регистрация, верификация или идентификация), может меняться в зависимости от используемой технологии и метода реализации, и интерфейс пользователя для регистрации обычно отличается от интерфейсов для верификации и идентификации. В первом случае интерфейс пользователя обычно включает в себя больше взаимодействий и является наиболее трудоемким, а во втором случае является наиболее простым и быстрым в использовании. Таким образом, стандартизация интерфейсов пользователя для биометрической регистрации, верификации или идентификации представляет собой сложную задачу в том случае, если необходимо предоставить стандартизированные механизмы или достаточную гибкость. Интерфейс ГИП БиоАПИ предоставляет этот стандарт с соответствующей гибкостью.

Большинство текущих реализаций ПБУ содержат встроенный интерфейс пользователя (на самом деле это требование БиоАПИ для соответствующей реализации ПБУ), таким образом, приложению не требуется знать все подробности о взаимодействии между ПБУ и пользователем, предоставившим биометрический образец. Однако БиоАПИ также дает возможность приложению контролировать «вид и поведение» интерфейса пользователя, разрешая приложению предоставить обработчик событий ГИП, который вызывает инфраструктура БиоАПИ в течение определенных операций в ответ на обратные вызовы, поступающие от ПБУ. Ниже приведены некоторые преимущества управляемого приложением ГИП над ГИП управляемым ПБУ:

а) приложение может использовать графические изображения и текст (возможно в отдельном окне) для предоставления пользователю обратной связи, задач или руководств относительно текущей биометрической операции;

b) изображения могут демонстрироваться таким образом, чтобы соответствовать виду и поведению отдельного приложения и находиться в главном окне приложения или в отдельном окне, по усмотрению приложения; и

c) приложение может применять язык человека, который может не поддерживаться ПБУ, так как (если используется управляемый приложением ГИП) большинство взаимодействий посредством обратных вызовов между ПБУ и приложением (через инфраструктуру БиоАПИ) просто идентифицируют семантику, которая должна быть предоставлена пользователю; язык человека, применяемый для предоставления сообщения, является проблемой для приложения (несмотря на то, что событие выполнения ГИП также позволяет ПБУ предоставлять полутоновые и цветные изображения).

Уведомления о событиях выбора и изменения состояния ГИП применяются для управления сбором образцов и для указания приложению на изменение в состоянии (в процессе операций регистрации, верификации или идентификации). Приложение вызывает функцию ***BioAPI_SubscribeToGUIEvents*** для управления ГИП самостоятельно вместо того, чтобы позволять ПБУ управлять ГИП.

События выбора и изменения состояния ГИП генерируются ПБУ только в процессе выполнения операции, инициализированной вызовом приложением любой из следующих функций БиоАПИ:

- BioAPI_Capture;
- BioAPI_Enroll;
- BioAPI_Verify;
- BioAPI_Identify.

События выполнения ГИП могут быть сгенерированы ПБУ в процессе того, как ПБУ выполняет операцию, инициализированную вызовом приложения любой из следующих функций БиоАПИ:

- BioAPI_Capture;
- BioAPI_Process;
- BioAPI_CreateTemplate;
- BioAPI_VerifyMatch;
- BioAPI-IdentifyMatch;
- BioAPI_Enroll;
- BioAPI_Verify;
- BioAPI_Identify.

Приложение получит уведомления о событиях ГИП, которые предоставляют информацию, связанную с ГИП, в течение выполнения операций получения данных, регистрации, верификации или идентификации. В таблице С.1 представлен список событий ГИП, которые могут произойти.

Таблица С.1 — События ГИП

Событие ГИП	Описание
Событие выбора	Данное событие ГИП генерируется ПБУ перед запуском и после окончания цикла подопераций в процессе операций получения данных, верификации, идентификации и регистрации. В начале цикла подопераций событие выбора ГИП позволяет приложению определить (в его ответе на обратный вызов уведомления) подтип (см. ИСО/МЭК 19785-1, 6.5.7) или подтипы образцов, которые должны быть захвачены (например, для пальца: левый/ правый, указательный/ средний/ безымянный/ мизинец) при помощи подоперации(й) захвата в рамках цикла подопераций, который скоро будет запущен. Некоторые ПБУ способны захватывать одновременно несколько подтипов или экземпляров биометрической модальности (например, все пять пальцев или обе радужные оболочки глаза) за одну подоперацию захвата, поэтому в ответ на многоэкземплярный захват предоставляется битовая маска. В конце цикла подопераций событие выбора ГИП сообщает приложению о завершении цикла подопераций (возможно предоставляя дополнительную информацию) и позволяет приложению указать на то, должен ли ПБУ считать всю операцию завершенной (с возвратом приложению BioAPI_OK или кодом ошибки), прервать операцию (с возвратом приложению кода ошибки) или сбросить все текущие результаты и запустить новый цикл подопераций (без выхода из первоначального вызова функции ПБУ)
Событие изменения состояния	Данное событие генерируется ПБУ перед запуском и после завершения каждой подоперации операции. Определены следующие пять подопераций: - захват данных: захватывает один или несколько экземпляров модальности при помощи модуля сканера ПБУ и создает образец промежуточного уровня обработки с целью его последующей верификации, идентификации или регистрации; - обработка: обрабатывает промежуточный образец с целью его последующей верификации или идентификации и преобразовывает в обработанный образец;

Продолжение таблицы С.1

Событие ГИП	Описание
Событие изменения состояния	- создание шаблона: обрабатывает промежуточный образец с целью его последующей регистрации и преобразовывает его в контрольный шаблон, возможно после выбора «лучшего» промежуточного образца из набора; - верификация-сопоставление: выполняет верификацию-сопоставление образца, обработанного с целью его последующей верификации, и контрольного шаблона; в контексте операции регистрации, это также называется тест-верификация; - идентификация-сопоставление: выполняет идентификацию-сопоставление образца, обработанного с целью его последующей идентификации, и множества контрольных шаблонов. Пять подопераций имеют те же наименования, что и пять функций БиоАПИ, потому что они выполняют, в сущности, те же задания, хотя на другом абстрактном уровне. К примеру, для BioSPI_Process существует единственная подоперация обработки, также для BioSPI_CreateTemplate (единственная подоперация создания шаблона), BioSPI_VerifyMatch (единственная подоперация верификации-сопоставления) и BioSPI_IdentifyMatch (единственная подоперация идентификации-Агсопоставления). Однако для более сложных операций (BioSPI_Capture, BioSPI_Verify, BioSPI_Identify и BioSPI_Enroll) существует цикл подопераций, включающий в себя одну или более подопераций захвата данных, за которыми, возможно, следуют другие подоперации. Полный цикл может быть повторен один раз или более без выхода из операции. Если ПБУ предоставляет тип регистрации Многократный-Захват функции BioAPI_GUI_SELECT_EVENT_HANDLER, то цикл подопераций для BioSPI_Enroll состоит из подопераций многократного захвата, за которыми следует подоперация создания шаблона. Если ПБУ предоставляет тип регистрации Тестировать-Верифицировать, то цикл подопераций для BioSPI_ENROLL состоит из захвата с целью его дальнейшей регистрации, захвата с целью его дальнейшей верификации, создания шаблона, обработки и верификации-сопоставления (тест верификации)

Окончание таблицы С.1

Событие ГИП	Описание
Событие выполнения	ПБУ может сгенерировать данное событие в процессе любой подоперации в рамках операции захвата данных, обработки, создания шаблона, верификации-сопоставления, идентификации-сопоставления, верификации, идентификации или регистрации для сообщения приложению о выполнении (степени выполнения, завершенности) подоперации. Уведомление о событии выполнения ГИП для подоперации получения данных (захвата) может включать в себя потоковые данные (например, поток видео в реальном времени). Многократные уведомления о событии выполнения ГИП для длительных подопераций идентификации-сопоставления могут использоваться для сообщения пользователю о проценте выполнения подоперации идентификации-сопоставления

ПБУ управляется конечным механизмом, связанным с операцией, и предоставляет информацию об изменении состоянии, текстовые сообщения и (в соответствующих случаях) битовые изображения путем вызова дескриптора инфраструктуры уведомления о событии ГИП, который в свою очередь отвечает вызовом дескриптора уведомления о событии ГИП приложения. После демонстрации изображения пользователю (или оператору, управляющему биометрическим процессом), приложение определяет, продолжить ли вызывать первоначальную функцию, отменить ее или изменить путь, ответив обратному вызову уведомления о событии ГИП.

Событие выполнения ГИП может быть применено для предоставления потоковых данных приложению (предназначенных для демонстрации пользователю или оператору) в форме последовательности битовых изображений. В течение подоперации идентификации-сопоставления событие выполнения ГИП может быть применено для предоставления пользователю степени выполнения идентификации. Генерация события выполнения ГИП является прерогативой ПБУ, который указывает в своем реестре компонентов информацию о поддержке реализации данной возможности.

Функция ***BioAPI_SubscribeToGUIEvents*** подписывает приложение на события выбора, изменения состояния и выполнения ГИП, в то вре-

мя как функция ***BioAPI_Unsubscribe FromGUIEvents*** удаляет существующую подписку на событие ГИП. Все ПБУ, которые реализуют возможность управления ГИП приложением, необходимы для поддержки соответствующих функций ПБУ ***BioSPI_SubscribeToGUIEvents*** и ***BioSPI_UnsubscribeFromGUIEvents.***

Взаимодействия между биометрическим приложением и ПБУ, поддерживающим регистрацию, определяется типом регистрации, предоставленным ПБУ. Далее определены следующие два типа регистрации:

a) Тестировать-Верифицировать: В процессе операции регистрации (инициированной вызовом ***BioAPI_Enroll***) ПБУ, предоставляющий данный тип регистрации, выполняет:

1) подоперацию захвата данных для создания кандидата на контрольный шаблон;

2) вторую подоперацию захвата данных для получения образца для тестирования;

3) подоперацию по созданию шаблона (по первому образцу);

4) подоперацию обработки (по второму образцу); и

5) подоперацию верификации-сопоставления для определения того, совпадает ли образец для тестирования и кандидат на контрольный шаблон.

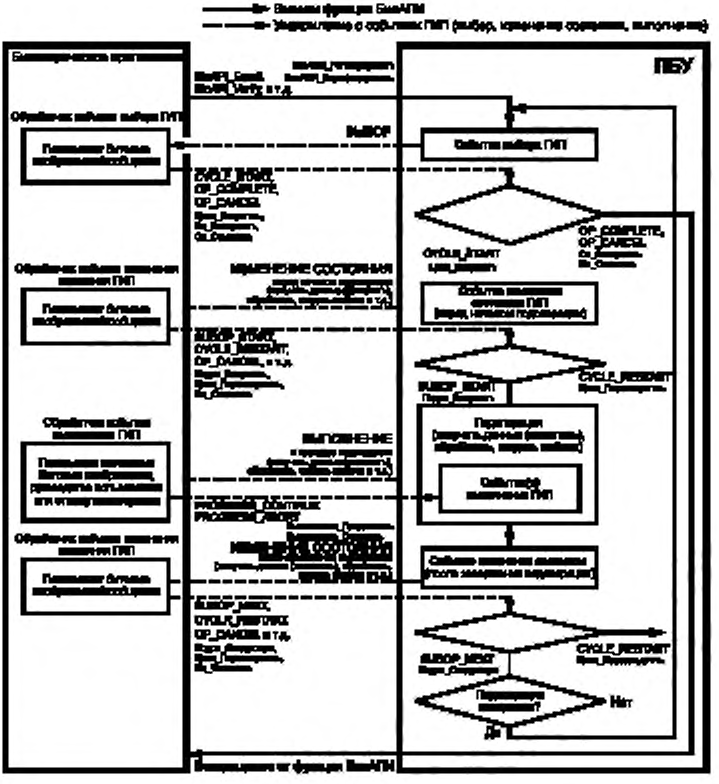
Кандидат на контрольный шаблон принимается только в том случае, когда операция верификации проведена успешно. В противном случае приложение может решить, должен ли цикл быть повторен, либо операция регистрации должна завершиться с ошибкой;

b) Многократный-Захват: ПБУ, предоставляющий данный тип регистрации, точно установил критерии соответствия образца для создания контрольного шаблона. В процессе операции регистрации ПБУ выполняет одну или более подопераций захвата данных в рамках определенного заранее ограничения (*MaxNumEnrollInstances*, см. 7.60) с целью получения образца, удовлетворяющего всем критериям.

Примечание – ПБУ, выполняющий (вероятно) несколько захватов с целью получения образца, используемого в качестве кандидата на регистрацию, и далее захватывающий следующий образец для сопоставления с кандидатом (сочетание Тестировать-Верифицировать и Многократного-Захвата) не поддерживается в версиях БиоАПИ 2.0 или 2.1.

ПБУ предоставляет свой тип регистрации в качестве входного параметра функции BioAPI_GUI_SELECT_EVENT_HANDLER, таким образом делая его доступным для приложения в начале каждой операции регистрации.

На рисунке С.2а показана обобщенная схема процесса реализации для ГИП, управляемого приложением.



Примечание — В действительной реализации между приложением и ПБУ находится инфраструктура БиоАПИ. Инфраструктура БиоАПИ управляет всеми вызовами и уведомлением о событиях БиоАПИ.

Рисунок С.2а — Пример обобщенного потока процесса реализации ГИП

С.7.2 Пример последовательности уведомлений о событиях ГИП для операции Регистрации (тип регистрации — Тестировать-Верифицировать, без ошибок и перезапусков)

Пример последовательности уведомлений о событиях ГИП представлен в таблице С.2.

Таблица С.2

Событие ГИП	Входные параметры	Выходные параметры
выбор	Операция = **BioAPI_GUI_ OPERATION_ ENROLL** Момент = **BioAPI_GUI_ MOMENT_BEFORE_START** Код Результата = н/д МаксЧислоОбразцовРегистрации = 1 ВыбираемыеЭкземпляры = левый указательный палец, правый указательный палец ЗахваченныеЭкземпляры = н/д	Ответ = **BioAPI_GUI_ RESPONSE_CYCLE_ START** ВыбранныеЭкземпляры = левый указательный палец
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_ GUI_SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_ BEFORE_ START** Код Результата = н/д ИндексОбразцаРегистрации = 1	Ответ = **BioAPI_GUI_ RESPONSE_SUBOP_ START** ИндексОбразцаРегистрацииНаПерезахват = н/д
выполнение	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_ SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE ENROLL** Момент = **BioAPI_GUI_ MOMENT_DURING** СтепеньВыполненияПодоперации = 20	Ответ = **BioAPI_ GUI_ RESPONSE_PROGRESS_ CONTINUE**

Продолжение таблицы С.2

Событие ГИП	Входные параметры	Выходные параметры
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 50	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = 1	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CREATETEMPLATE** Назначение = н/д Момент = **BioAPI_GUI_MOMENT_BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_START** ИндексОбразцаРегистрацииНаПерезахват = н/д

Продолжение таблицы С.2

Событие ГИП	Входные параметры	Выходные параметры
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CREATETEMPLATE** Назначение = н/д Момент = **BioAPI_GUI_MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_VERIFY** Момент = **BioAPI_GUI_MOMENT_BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_START** ИндексОбразцаРегистрацииНаПерезахват = н/д
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_VERIFY** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации =10	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**

Продолжение таблицы С.2

Событие ГИП	Входные параметры	Выходные параметры
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_VERIFY** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 40	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_VERIFY** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 90	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_VERIFY** Момент = **BioAPI_GUI_MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д

Продолжение таблицы С.2

Событие ГИП	Входные параметры	Выходные параметры
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_ SUBOPERATION_ PROCESS** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ SUBOP_START** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_ SUBOPERATION_ PROCESS** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ SUBOP_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_ SUBOPERATION_ VERIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_ BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ SUBOP_START** ИндексОбразцаРегистрацииНаПерезахват = н/д

Окончание таблицы С.2

Событие ГИП	Входные параметры	Выходные параметры
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_VERIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д
выбор	Операция = **BioAPI_GUI_OPERATION_ENROLL** Момент = **BioAPI_GUI_MOMENT_AFTER_END** Код Результата = **BioAPI_OK** МаксЧислоОбразцовРегистрации = 1 ВыбираемыеЭкземпляры = левый указательный палец, правый указательный палец ЗахваченныеЭкземпляры = левый указательный палец	Ответ = **BioAPI_GUI_RESPONSE_OP_COMPLETE** ВыбранныеЭкземпляры = н/д

С.7.3 Пример последовательности уведомлений о событиях ГИП для операции Регистрации (тип регистрации — Многократный-Захват, без ошибок и перезапусков, один перезахват)

Пример последовательности уведомлений о событиях ГИП представлен в таблице С.3.

Таблица С.3

Событие ГИП	Входные параметры	Выходные параметры
выбор	Операция = **BioAPI_GUI_OPERATION_ENROLL** Момент = **BioAPI_GUI_ MOMENT_BEFORE_START** Код Результата = н/д МаксЧислоОбразцовРегистрации = 3 ВыбираемыеЭкземпляры = левый указательный палец, правый указательный палец ЗахваченныеЭкземпляры = н/д	Ответ = **BioAPI GUI_ RESPONSE_CYCLE_ START** ВыбранныеЭкземпляры = левый указательный палец
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_ GUI_SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE_ ENROLL** Момент = **BioAPI_GUI_MOMENT_ BEFORE_ START** Код Результата = н/д ИндексОбразцаРегистрации = 1	Ответ = **BioAPI_GUI_ RESPONSE_SUBOP_ START** ИндексОбразцаРегистрацииНаПерезахват = н/д
выполнение	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_ SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE ENROLL** Момент = **BioAPI_GUI_ MOMENT_ DURING** СтепеньВыполненияПодоперации = 18	Ответ = **BioAPI_ GUI_ RESPONSE_PROGRESS_ CONTINUE**

Продолжение таблицы С.3

Событие ГИП	Входные параметры	Выходные параметры
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 65	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = 1	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = 2	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_START** ИндексОбразцаРегистрацииНаПерезахват = н/д

Продолжение таблицы С.3

Событие ГИП	Входные параметры	Выходные параметры
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 30	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 80	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**
выполнение	Операция = **BioAPI_GUI_OPERATION ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 95	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**

Продолжение таблицы С.3

Событие ГИП	Входные параметры	Выходные параметры
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = 2	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = 3	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_START** ИндексОбразцаРегистрацииНаПерезахват = н/д
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 15	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**

Продолжение таблицы С.3

Событие ГИП	Входные параметры	Выходные параметры
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 70	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**
выполнение	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 90	Ответ = **BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE**
изменение состояния	Операция = **BioAPI_GUI_OPERATION_ENROLL** Подоперация = **BioAPI_GUI SUBOPERATION_CAPTURE** Назначение = **BioAPI_PURPOSE_ENROLL** Момент = **BioAPI_GUI_MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = 3	Ответ = **BioAPI_GUI_RESPONSE_SUBOP_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д

Продолжение таблицы С.3

Событие ГИП	Входные параметры	Выходные параметры
изменение состояния	Операция = **BioAPI_ GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_ CREATETEMPLATE** Назначение = н/д Момент = **BioAPI_GUI_MOMENT_ BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ SUBOP_START** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_ CREATETEMPLATE** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_ AFTER_END** Код Результата = **BioAPI-ERR_LOW_QUALITY_ REFERENCE_TEMPLATE** ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ RECAPTURE** ИндексОбразцаРегистрацииНаПерезахват = 2
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION CAPTURE** Назначение = **BioAPI_PURPOSE_ ENROLL** Момент = **BioAPI_GUI_MOMENT_ BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = 2	Ответ = **BioAPI_GUI_RESPONSE_ SUBOP_START** ИндексОбразцаРегистрацииНаПерезахват = н/д

Продолжение таблицы С.3

Событие ГИП	Входные параметры	Выходные параметры
выполнение	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE_ ENROLL** Момент = **BioAPI_GUI_ MOMENT_DURING** СтепеньВыполненияПодоперации = 45	Ответ = **BioAPI_GUI_RESPONSE_ PROGRESS_CONTINUE**
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE_ ENROLL** Момент = **BioAPI_GUI_ MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = 2	Ответ = **BioAPI_GUI_RESPONSE_ NEXT_SUBOP** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение состояния	Операция = **BioAPI_ GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_ CREATETEMPLATE** Назначение = н/д Момент = **BioAPI_GUI_MOMENT_ BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ START_SUBOP** ИндексОбразцаРегистрацииНаПерезахват = н/д

Окончание таблицы С.3

Событие ГИП	Входные параметры	Выходные параметры
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_ENROLL** Подоперация = **BioAPI_GUI_SUBOPERATION_ CREATETEMPLATE** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ NEXT_SUBOP** ИндексОбразцаРегистрацииНаПерезахват = н/д
выбор	Операция = **BioAPI_ GUI_ OPERATION_ENROLL** Момент = **BioAPI_GUI_ MOMENT_AFTER_END** Код Результата = **BioAPI_OK** МаксЧислоОбразцовРегистрации = 1 ВыбираемыеЭкземпляры = левый указательный палец, правый указательный палец ЗахваченныеЭкземпляры = левый указательный палец	Ответ = **BioAPI_GUI_RESPONSE_ OP_COMPLETE** ВыбранныеЭкземпляры = н/д
выбор	Операция = **BioAPI_ GUI_ OPERATION_INDENTIFY** Момент = **BioAPI_GUI_ MOMENT_BEFORE_START** Код Результата = н/д МаксЧислоОбразцовРегистрации = н/д ВыбираемыеЭкземпляры = правый ЗахваченныеЭкземпляры = н/д	Ответ = **BioAPI_GUI_ RESPONSE_START_CYCLE_** ВыбранныеЭкземпляры = правый

С.7.4 Пример последовательности уведомлений о событиях ГИП для операции Идентификации (без ошибок и перезапусков)

Пример последовательности уведомлений о событиях ГИП представлен в таблице С.4.

Таблица С.4

Событие ГИП	Входные параметры	Выходные параметры
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_INDENTIFY** Подоперация = **BioAPI_ GUI_SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE_INDENTIFY** Момент = **BioAPI_GUI_MOMENT_ BEFORE_ START** Код Результата = н/д ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_ RESPONSE_START_ SUBOP** ИндексОбразцаРегистрацииНаПерезахват = н/д
выполнение	Операция = **BioAPI_GUI_ OPERATION_INDENTIFY** Подоперация = **BioAPI_GUI_ SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE_ INDENTIFY** Момент = **BioAPI_GUI_ MOMENT_DURING** СтепеньВыполненияПодоперации = 32	Ответ = **BioAPI_ GUI_ RESPONSE_ CONTINUE_SUBOP**
выполнение	Операция = **BioAPI_GUI_ OPERATION_INDENTIFY** Подоперация = **BioAPI_GUI_ SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE_ INDENTIFY** Момент = **BioAPI_GUI_ MOMENT_DURING** СтепеньВыполненияПодоперации = 70	Ответ = **BioAPI_ GUI_ RESPONSE_ CONTINUE_SUBOP**

Продолжение таблицы С.4

Событие ГИП	Входные параметры	Выходные параметры
изменение состояния	Операция = **BioAPI_GUI_ OPERATION_IDENTIFY** Подоперация = **BioAPI_GUI_SUBOPERATION_ CAPTURE** Назначение = **BioAPI_PURPOSE_ IDENTIFY** Момент = **BioAPI_GUI_ MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = 1	Ответ = **BioAPI_GUI_RESPONSE_ NEXT_SUBOP** ИндексОбразцаРегистрацииНаПерезахват = н/д
изменение состояния	Операция = **BioAPI_ GUI_ OPERATION_IDENTIFY** Подоперация = **BioAPI_GUI_ SUBOPERATION_IDENTIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_BEFORE_START** Код Результата = н/д ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ STAR_SUBOP** ИндексОбразцаРегистрацииНаПерезахват = н/д
выполнение	Операция = **BioAPI_ GUI_ OPERATION_IDENTIFY** Подоперация = **BioAPI_GUI SUBOPERATION_IDENTIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_MOMENT_DURING** СтепеньВыполненияПодоперации = 9	Ответ = **BioAPI_GUI_RESPONSE_ PROGRESS_CONTINUE**

Продолжение таблицы С.4

Событие ГИП	Входные параметры	Выходные параметры
выполнение	Операция = **BioAPI_ GUI_ OPERATION_IDENTIFY** Подоперация = **BioAPI_GUI SUBOPERATION_IDENTIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_DURING** СтепеньВыполненияПодоперации = 30	Ответ = **BioAPI_GUI_RESPONSE_ PROGRESS_CONTINUE**
выполнение	Операция = **BioAPI_ GUI_ OPERATION_IDENTIFY** Подоперация = **BioAPI_GUI_ SUBOPERATION_IDENTIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_DURING** СтепеньВыполненияПодоперации = 64	Ответ = **BioAPI_GUI_RESPONSE_ PROGRESS_CONTINUE**
выполнение	Операция = **BioAPI_ GUI_ OPERATION_IDENTIFY** Подоперация = **BioAPI_GUI SUBOPERATION_IDENTIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_DURING** СтепеньВыполненияПодоперации = 83	Ответ = **BioAPI_GUI_RESPONSE_ PROGRESS_CONTINUE**

Окончание таблицы С.4

Событие ГИП	Входные параметры	Выходные параметры
выполнение	Операция = **BioAPI_ GUI_ OPERATION_IDENTIFY** Подоперация = **BioAPI_GUI_ SUBOPERATION_IDENTIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_DURING** СтепеньВыполненияПодоперации = 98	Ответ = **BioAPI_GUI_RESPONSE_ PROGRESS_CONTINUE**
изменения состояния	Операция = **BioAPI_ GUI_ OPERATION_IDENTIFY** Подоперация = **BioAPI_GUI_ SUBOPERATION_IDENTIFYMATCH** Назначение = н/д Момент = **BioAPI_GUI_ MOMENT_AFTER_END** Код Результата = **BioAPI_OK** ИндексОбразцаРегистрации = н/д	Ответ = **BioAPI_GUI_RESPONSE_ SUBOR_NEXT** ИндексОбразцаРегистрацииНаПерезахват = н/д
выбор	Операция = **BioAPI_ GUI_ OPERATION_IDENTIFY** Момент = **BioAPI_GUI_ MOMENT_AFTER_END** Код Результата = **BioAPI_OK** МаксЧислоОбразцовРегистрации = н/д ВыбираемыеЭкземпляры = правый ЗахваченныеЭкземпляры = правый	Ответ = **BioAPI_GUI_RESPONSE_ OP_COMPLETE** ВыбранныеЭкземпляры = н/д

Пункт С.8.4. Первый абзац изложить в новой редакции:

«Существует множество способов взаимодействия биометрических компонентов и компонентов смарт-карты (программные и аппаратные средства) для выполнения операции СНК. Один из потоков процесса высокого уровня для такой операции изображен на рисунке С.3».

Приложение D дополнить подразделом — D.3:

«D.3 Последовательность вызовов с ГИП (БиоАПИ 2.1)

Примеры в пунктах D.3.1, D.3.2 применимы только при использовании БиоАПИ версии 2.1.

D.3.1 Регистрация с ГИП

Если ПБУ поддерживает функцию BioAPI_SunscribeToGUIEvents, и тип регистрации, предоставленный ПБУ, является Тестировать-Верифицировать, то последовательность вызовов при регистрации может быть следующая:

```
int ApplicationControlledEnrollFunction()
    {
        BioAPI_RETURN bioReturn;
        BioAPI_BIR_HANDLE EnrolledTemplate;
        BioAPI_HANDLE BSPHandle;
        BioAPI_GUI_SELECT_EVENT_HANDLER  EnrollGUI-
            SelectEventHandler;
        BioAPI_GUI_STATE_EVENT_HANDLER EnrollGUIState-
            EventHandler;
        BioAPI_GUI_PROGRESS_EVENT_HANDLER EnrollGUI-
            ProgressEventHandler;
        // Задание обработчиков событий ГИП.
        bioReturn = BioAPI_SubscribeToGUIEvents(
            NULL,
            NULL,
            &BSPHandle,
            EnrollGUISelectEventHandler,
            NULL,
            EnrollGUIStateEventHandler,
            NULL,
            EnrollGUIProgressEventHandler,
            NULL);
        if(bioReturn != BioAPI_OK)
        {
          printf("Код Ошибки БиоАПИ: %d\n", bioReturn);
          return 0;
        }
```

```
        // При вызове функции BioAPI_Enroll события ГИП
        // будут вызваны перед возвращением значения этой функции.
        bioReturn = BioAPI_Enroll(
                BSPHandle,
                BioAPI_PURPOSE_ENROLL
                NULL,
                &EnrolledTemplate,
                NULL,-1, NULL, NULL);
        // После того, как функция BioAPI_Enroll возвратила значе-
        ние,
        // приложение может удалить записи обратного вызова путем
        // вызова функции BioAPI_UnsubscribeFromGUIEvents.
        BioAPI_UnsubscribeFromGUIEvents (
                NULL,
                NULL,
                &BSPHandle,
                EnrollGUISelectEventHandler,
                NULL,
                EnrollGUIStateEventHandler,
                NULL,
                EnrollGUIProgressEventHandler,
                NULL);
        if(bioReturn != BioAPI_OK)
        {
                printf("Код Ошибки БиоАПИ: %d\n", bioReturn);
                return 0;
        }
        return 1;
}
BIOAPI_RETURN CALLBACK EnrollGUISelectEventHandler
        (const BioAPI_UUID *BSPUuid,
        BioAPI_UNIT_ID UnitID,
        const BioAPI_HANDLE *BSPHandle,
        uint32_t EnrollType,
        const void *GUISelectEventHandlerCtx,
        BioAPI_GUI_OPERATION Operation,
        BioAPI_GUI_MOMENT Moment,
        BioAPI_RETURN ResultCode,
        int32_t MaxNumEnrollSamples,
        BioAPI_BIR_SUBTYPE_MASK SelectableInstances,
        BioAPI_BIR_SUBTYPE_MASK*SelectedInstances,
```

```
        BioAPI_BIR_SUBTYPE_MASK CapturedInstances,
        const uint8_t *Text,
        BioAPI_GUI_RESPONSE *Response)
{
// Показать ГИП пользователю для выбора конкретного
// биометрического подтипа с целью его последующей регистрации.
// Задать параметр SelectedInstances, основанный на подтипе(ах),
// выбранном(ых) пользователем.
// Если событие выбора ГИП сгенерировано перед циклом
подопераций
// (Moment = BioAPI_GUI_MOMENT_BEFORE_START), то
необходимо задать
// параметру Response значение BioAPI_GUI_RESPONSE_CYCLE_
START,
// если пользователь хочет приступить к регистрации, или
// значение BioAPI_GUI_RESPONSE_OP_CANCEL, если
// пользователь хочет прервать регистрацию.
// Если событие выбора ГИП сгенерировано после цикла подопераций
// (Moment = BioAPI_GUI_MOMENT_AFTER END), то необходимо
задать
// параметру Response значение BioAPI_GUI_RESPONSE OP_
COMPLETE,
// если пользователь удовлетворен регистрацией, или
// значение BioAPI_GUI_RESPONSE_CYCLE_RESTART, если
// пользователь хочет повторить цикл, или
// значение BioAPI_GUI_RESPONSE_OP_CANCEL, если
// пользователь хочет прервать регистрацию.
        return BioAPI_OK;
}
BioAPI_RETURN CALLBACK EnrollGUIStateEventHandler
        (const BioAPI_UUID *BSPUuid,
        BioAPI_UNIT_ID UnitID,
        const BioAPI_HANDLE *BSPHandle,
        const void *GUIStateEventHandlerCtx,
        BioAPI_GUI_OPERATION Operation,
        BioAPI_GUI_SUBOPERATION Suboperation,
        BioAPI_BIR_PURPOSE Purpose,
        BioAPI_GUI_MOMENT Moment,
        BioAPI_RETURN ResultCode,
        int32_t EnrollSampleIndex,
        const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
```

```
        const uint8_t *Text,
        BioAPI_GUI_RESPONSE *Response,
        int32_t *EnrollSampleIndexToRecapture)
{
// Показать ГИП, основанный на значениях параметров
// Operation, Suboperation, Purpose, Moment, и
// ResultCode, предоставленных ПБУ. Если Текст (Text)
// был предоставлен, он может быть отображен.
        switch(Suboperation) {
        case BioAPI_GUI_SUBOPERATION_CAPTURE:
            switch(Purpose) {
            case BioAPI_PURPOSE_ENROLL:
            case BioAPI_PURPOSE_ENROLL_FOR_VERIFICATION_ONLY:
            case BioAPI_PURPOSE_ENROLL_FOR_
            IDENTICATION_ONLY:
                switch(Moment) {
                caseBioAPI_GUI_MOMENT_BEFORE_START:
                    // Пример сообщения: "Нажмите кнопку [Захват]
                    // для начала захвата изображения".
                    break;
                case BioAPI_GUI_MOMENT_AFTER_END:
                    // Пример сообщения 1: "Захват прошел
                    успешно".
                    // Пример сообщения 2: "Захват не удался".
                    break;
                }
                break;
            case BioAPI_PURPOSE_VERIFY:
               switch(Moment) {
               case BioAPI_GUI_MOMENT_BEFORE_START:
                    // Пример сообщения: "Нажмите кнопку [Захват],
                    чтобы
                    //начать захват изображения для операции
                    верификации".
                    break;
               case BioAPI_GUI_MOMENT_AFTER_END:
                    // Пример сообщения 1: "Захват прошел
                    успешно".
                    // Пример сообщения 2: "Захват не удался".
                    break;
```

```
                }
                break;
            }
            break;
        // другие подоперации
        }
        return BioAPI_OK;
    }
    BioAPI_RETURN CALLBACK EnrollGUIProgressEventHandler
        (const BioAPI_UUID *BSPUuid,
        BioAPI_UNIT_ID UnitID,
        const BioAPI_HANDLE *BSPHandle,
        const void *GUIProgressEventHandlerCtx,
        BioAPI_GUI_OPERATION Operation,
        BioAPI_GUI_SUBOPERATION Suboperation,
        BioAPI_BIR_PURPOSE Purpose,
        BioAPI_GUI_MOMENT Moment,
        uint8_t SuboperationProgress,
        const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
        const uint8_t *Text,
        BioAPI_GUI_RESPONSE *Response)
    {
        // Вывести потоковые данные на экран.
    }
```

Как только шаблон возвращен, он может быть помещен в хранилище данных для извлечения в дальнейшем с целью верификации (см. D.2).

D.3.2 Выполнение верификации с ГИП

Если ПБУ поддерживает функцию BioAPI_SubscribeToGUIEvents, то последовательность вызовов при верификации с помощью BioAPI_Verify может быть следующей:

```
int ApplicationControlledVerifyFunction ()
{
    BioAPI_INPUT_BIR birEnroll;
    BioAPI_BIR_HEADER birHeader;
    int MaxFMR, AchievedFMR;
    BioAPI_BOOL bPrecedence, bResponse;
    BioAPI_HANDLE BSPHandle;
    BioAPI_GUI_SELECT_EVENT_HANDLER VerifyGUISelectEventHandler;
    BioAPI_GUI_STATE_EVENT_HANDLER VerifyGUIStateEventHandler;
    BioAPI_GUI_PROGRESS_EVENT_HANDLER VerifyGUIProgressEvent-
        Handler;
```

```
// Задание событий ГИП.
bioReturn = BioAPI_SubscribeToGUIEvents(
        NULL,
        NULL,
        &BSPHandle,
        VerifyGUISelectEventHandler,
        NULL,
        VerifyGUIStateEventHandler,
        NULL,
        VerifyGUIProgressEventHandler,
        NULL);
// При вызове функции BioAPI_Verify обратный вызов изменения
// состояния ГИП будет вызван перед возвращением значения этой
функции.
MaxFMR=1;
bioReturn = BioAPI_Verify( BSPHandle,
        &MaxFMR,
        NULL,
        &bPrecedence,
        &birEnroll,
        NULL,
        &bResponse,
        &AchievedFMR,
        NULL,
        NULL,
        -1,
        NULL);
free(birEnroll.InputBIR.BIR);
// После того как функция BioAPI_Verify возвратила значение,
// приложение может удалить записи обратного вызова путем
// вызова функции BioAPI_UnsubscribeFromGUIEvents.
BioAPI_UnsubscribeFromGUIEvents (
        NULL,
        NULL,
        &BSPHandle,
        VerifyGUISelectEventHandler,
        NULL,
        VerifyGUIStateEventHandler,
        NULL,
        VerifyGUIProgressEventHandler,
        NULL);
```

```
if(bioReturn != BioAPI_OK)
{
        printf("Код Ошибки БиоАПИ: %d\n", bioReturn);
        return 0;
}
else if(bResponse != BioAPI_TRUE)
{
        printf("Нет Совпадений\n");
        return 0;
}
else
      {
        printf("Совпадение найдено\n");
        return 1;
      }
}
BIOAPI_RETURN CALLBACK VerifyGUISelectEventHandler
        (const BioAPI_UUID *BSPUuid,
        BioAPI_UNIT_ID UnitID,
        const BioAPI_HANDLE *BSPHandle,
        uint32_t EnrollType,
        const void *GUISelectEventHandlerCtx,
        BioAPI_GUI_OPERATION Operation,
        BioAPI_GUI_MOMENT Moment,
        BioAPI_RETURN ResultCode,
        int32_t MaxNumEnrollSamples,
        BioAPI_BIR_SUBTYPE_MASK SelectableInstances,
        BioAPI_BIR_SUBTYPE_MASK *SelectedInstances,
        BioAPI_BIR_SUBTYPE_MASK CapturedInstances,
        const uint8_t *Text,
        BioAPI_GUI_RESPONSE *Response)
{
// Показать ГИП пользователю для выбора конкретного
// биометрического подтипа с целью его последующей регистрации.
// Задать параметр SelectedInstances, основанный на подтипе(ах),
// выбранном(ых) пользователем.
// Если событие выбора ГИП сгенерировано перед циклом
подопераций
// (Moment = BioAPI_GUI_MOMENT_BEFORE_START), то
необходимо задать
```

```
// параметру Response значение BioAPI_GUI_RESPONSE_ CYCLE_
START, если
// пользователь хочет приступить к регистрации, или
// значение BioAPI_GUI_RESPONSE_OP_CANCEL, если
// пользователь хочет прервать регистрацию.
// Если событие выбора ГИП сгенерировано после цикла подопераций
// (Moment = BioAPI_GUI_MOMENT_AFTER_END), то необходимо
задать
// параметру Response значение BioAPI_GUI_RESPONSE_OP_
COMPLETE, если
// пользователь удовлетворен регистрацией, или
// значение BioAPI_GUI_RESPONSE_CYCLE_RESTART, если
// пользователь хочет повторить цикл, или
// значение BioAPI_GUI_RESPONSE_OP_CANCEL, если
// пользователь хочет прервать регистрацию.
        return BioAPI_OK;
}
BioAPI_RETURN CALLBACK VerifyGUIStateEventHandler
        (const BioAPI_UUID *BSPUuid,
        BioAPI_UNIT_ID UnitID,
        const BioAPI_HANDLE *BSPHandle,
        const void *GUIStateEventHandlerCtx,
        BioAPI_GUI_OPERATION Operation,
        BioAPI_GUI_SUBOPERATION Suboperation,
        BioAPI_BIR_PURPOSE Purpose,
        BioAPI_GUI_MOMENT Moment,
        BioAPI_RETURN ResultCode,
        int32_t EnrollSampleIndex,
        const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
        const uint8_t *Text,
        BioAPI_GUI_RESPONSE *Response,
        int32_t *EnrollSampleIndexToRecapture)
{
// Показать ГИП, основанный на значениях параметров
// Operation, Suboperation, Purpose, Moment, и
// ResultCode, предоставленных ПБУ. Если Текст (Text)
// был предоставлен, он может быть отображен.
        Switch (Suboperation){
        caseBioAPI_GUI_SUBOPERATION_CAPTURE:
```

```
            switch(Moment){
            case BioAPI_GUI MOMENT_BEFORE_START:
                  // Пример сообщения: "Нажмите кнопку [Захват]
                  // для начала захвата изображения".
                  break;
            caseBioAPI_GUI_MOMENT_AFTER_END:
                  // Пример сообщения 1: "Захват прошел успешно".
                  // Пример сообщения 2: "Захват не удался".
                  break;
            }
            break;
      case BioAPI_GUI_SUBOPERATION_VERIFYMATCH:
            switch(Moment) {
            caseBioAPI_GUI_MOMENT_BEFORE_START:
                  // Сообщения не требуются.
                  break;
            case BioAPI GUI_MOMENT AFTER_END:
                  // Пример сообщения 1: "Проверка прошла
                  успешно".
                  // Пример сообщения 2: "Проверка не удалась".
                  break;
            }
            break;
      // другие подоперации
      }
      return BioAPI_OK;
}
BioAPI_RETURN CALLBACK VerifyGUIProgressEventHandler
      (const BioAPI_UUID *BSPUuid,
      BioAPI_UNIT_ID UnitID,
      const BioAPI_HANDLE *BSPHandle,
      const void *GUIProgressEventHandlerCtx,
      BioAPI_GUI_OPERATION Operation,
      BioAPI_GUI_SUBOPERATION Suboperation,
      BioAPI_BIR_PURPOSE Purpose,
      BioAPI_GUI_MOMENT Moment,
      uint8_t SuboperationProgress,
      const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
```

```
        const uint8_t * Text,
        BioAPI_GUI_RESPONSE *Response)
{
        // Вывести потоковые данные на экран.
        return BioAPI_OK;
}  »
```

(ИУС № 9 2012 г.)

gost.e-ecolog.ru