ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО/МЭК 24708 – 2013

Информационные технологии

БИОМЕТРИЯ

Протокол межсетевого обмена БиоАПИ

ISO/IEC 24708:2008

Information technology – Biometrics – BioAPI Interworking Protocol

(IDT)

Издание официальное

Москва
Стандартинформ
2015

Предисловие

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 консультативной поддержке Ассоциации автоматической идентификации «ЮНИСКАН/GC1 РУС»

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 355 «Технологии автоматической идентификации и сбора данных и биометрия»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 06 сентября 2013 г. №990-ст

4 Настоящий стандарт идентичен международному стандарту ИСО/МЭК 24708:2008 «Информационные технологии. Биометрия. Протокол межсетевого обмена БиоАПИ» (ISO/IEC 24708:2008 «Information technology – Biometrics – BioAPI Interworking Protocol»).

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

5 ВВЕДЕН ВПЕРВЫЕ

6 Некоторые элементы настоящего стандарта могут быть объектами патентных прав. Организации ИСО и МЭК не несут ответственности за установление подлинности каких-либо или всех таких патентных прав

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 и на официальном сайте Федерального агентства по техническому регулированию и метрологии

© Стандартинформ, 2015

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
2 Нормативные ссылки
3 Термины и определения
4 Сокращения
5 Условные обозначения
6 Соответствие
7 Архитектура ПМО БиоАПИ
7.1 ПМО БиоАПИ-поддерживающие инфраструктуры
7.2 Сообщения ПМО БиоАПИ
7.3 Конечные точки ПМО БиоАПИ
7.4 Связи ПМО БиоАПИ
7.5 Привязки транспортного протокола
7.6 Создание и разрушение связей ПМО БиоАПИ
8 Уведомление об удаленных операциях графического интерфейса пользователя (ГИП)
9 Примеры возможных конфигураций системы
10 Форматы ПМО БиоАПИ
11 Идентификация конечных точек ПМО БиоАПИ, приложений и ПБУ
12 Краткий обзор обменов ПМО БиоАПИ
11.1 Обеспечение безопасности и конфиденциальности
11.2 Вызов приложением функций на удаленном ПБУ
11.3 Обращение приложения к функциям, не имеющим связанных с ними сообщений ПМО БиоАПИ
11.4 Операционные уведомления
13 Общие положения
14 Синтаксис сообщений ПМО БиоАПИ
15 Типы БиоАПИ и ПМО БиоАПИ
15.1 Целые числа
15.2 Символы строкового типа

15.3 Унифицированный идентификатор ресурса назначения конечных точек ПМО БиоАПИ
15.4 Тип BioAPI_BFP_LIST_ELEMENT
15.5 Тип BioAPI_BFP_SCHEMA
15.6 Тип BioAPI_BIR
15.7 Тип BioAPI_BIR_ARRAY_POPULATION
15.8 Тип BioAPI_BIR_BIOMETRIC_DATA_FORMAT
15.9 Тип BioAPI_BIR_BIOMETRIC_PRODUCT_ID
15.10 Тип BioAPI_BIR_DATA_TYPE
15.11 Тип BioAPI_BIR_HANDLE
15.12 Тип BioAPI_BIR_HEADER
15.13 Тип BioAPI_BIR_PURPOSE
15.14 Тип BioAPI_BIR_SECURITY_BLOCK_FORMAT
15.15 Тип BioAPI_BIR_SUBTYPE
15.16 Тип BioAPI_BIR_SUBTYPE_MASK
15.17 Тип BioAPI_BOOL
15.18 Тип BioAPI_BSP_SCHEMA
15.19 Тип BioAPI_CANDIDATE
15.20 Тип BioAPI_CATEGORY
15.21 Тип BioAPI_DATA
15.22 Тип BioAPI_DATE
15.23 Тип BioAPI_DB_ACCESS_TYPE
15.24 Тип BioAPI_DB_MARKER_HANDLE
15.25 Тип BioAPI_DB_HANDLE
15.26 Тип BioAPI_DBBIR_ID
15.27 Тип BioAPI_DTG
15.28 Тип BioAPI_ERROR_INFO
15.29 Тип BioAPI_EVENT
15.30 Тип BioAPI_EVENT_MASK
15.31 Тип BioAPI_FMR

15.32 Тип BioAPI_FRAMEWORK_SCHEMA
15.33 Тип BioAPI_GUI_BITMAP
15.34 Тип BioAPI_GUI_BITMAP_ARRAY
15.35 Тип BioAPI_GUI_EVENT_SUBSCRIPTION
15.36 Тип BioAPI_GUI_MOMENT
15.37 Тип BioAPI_GUI_ENROLL_TYPE
15.38 Тип BioAPI_GUI_OPERATION
15.39 Тип BioAPI_GUI_RESPONSE
15.40 Тип BioAPI_GUI_SUBOPERATION
15.41 Тип BioAPI_HANDLE
15.42 Тип BioAPI_IDENTIFY_POPULATION
15.43 Тип BioAPI_IDENTIFY_POPULATION_TYPE
15.44 Тип BioAPI_INDICATOR_STATUS
15.45 Тип BioAPI_INPUT_BIR
15.46 Тип BioAPI_INPUT_BIR_FORM
15.47 Тип BioAPI_OPERATIONS_MASK
15.48 Тип BioAPI_OPTIONS_MASK
15.49 Тип BioAPI_POWER_MODE
15.50 Тип BioAPI_QUALITY
15.51 Тип BioAPI_RETURN
15.52 Тип BioAPI_STRING
15.53 Тип BioAPI_TIME
15.54 Тип BioAPI_UNIT_ID
15.55 Тип BioAPI_UNIT_LIST_ELEMENT
15.56 Тип BioAPI_UNIT_SCHEMA
15.57 Тип BioAPI_UUID
15.58 Тип BioAPI_VERSION
16 Функции, определенные в БиоАПИ, и соответствующие сообщения ПМО БиоАПИ
16.1 Функция BioAPI_Init

16.2 Функция BioAPI_InitEndpoint
16.3 Функция BioAPI_Terminate
16.4 Функция BioAPI_LinkToEndpoint
16.5 Функция BioAPI_UnlinkFromEndpoint
16.6 Функция BioAPI_EnumFrameworks
16.7 Функция BioAPI_EnumBSPs
16.8 Функция BioAPI_EnumBFPs
16.9 Функция BioAPI_BSPLoad
16.10 Функция BioAPI_BSPUnload
16.11 Функция BioAPI_QueryUnits
16.12 Функция BioAPI_QueryBFPs
16.13 Функция BioAPI_BSPAttach
16.14 Функция BioAPI_BSPDetach
16.15 Функция BioAPI_EnableEvents
16.16 Функция BioAPI_EnableEventNotifcations
16.17 Функция BioAPI_ControlUnit
16.18 Функция BioAPI_Control
16.19 Функция BioAPI_FreeBIRHandle
16.20 Функция BioAPI_GetBIRFromHandle
16.21 Функция BioAPI_GetHeaderFromHandle
16.22 Функция BioAPI_SubscribeToGUIEvents
16.23 Функция BioAPI_UnsubscribeFromGUIEvents
16.24 Функция BioAPI_QueryGUIEventSubscriptions
16.25 Функция BioAPI_NotifyGUISelectEvent
16.26 Функция BioAPI_NotifyGUIStateEvent
16.27 Функция BioAPI_NotifyGUIProgressEvent
16.28 Функция BioAPI_RedirectGUIEvents
16.29 Функция BioAPI_UnredirectGUIEvents
16.30 Функция BioAPI_Capture
16.31 Функция BioAPI_CreateTemplate

16.32 Функция BioAPI_Process
16.33 Функция BioAPI_ProcessWithAuxBIR
16.34 Функция BioAPI_VerifyMatch
16.35 Функция BioAPI_IdentifyMatch
16.36 Функция BioAPI_Enroll
16.37 Функция BioAPI_Verify
16.38 Функция BioAPI_Identify
16.39 Функция BioAPI_Import
16.40 Функция BioAPI_PresetIdentifyPopulation
16.41 Функция BioAPI_Transform
16.42 Функция BioAPI_DbOpen
16.43 Функция BioAPI_DbClose
16.44 Функция BioAPI_DbCreate
16.45 Функция BioAPI_DbDelete
16.46 Функция BioAPI_DbSetMarker
16.47 Функция BioAPI_DbFreeMarker
16.48 Функция BioAPI_DbStoreBIR
16.49 Функция BioAPI_DbGetBIR
16.50 Функция BioAPI_DbGetNextBIR
16.51 Функция BioAPI_DbDeleteBIR
16.52 Функция BioAPI_CalibrateSensor
16.53 Функция BioAPI_SetPowerMode
16.54 Функция BioAPI_SetIndicatorStatus
16.55 Функция BioAPI_GetIndicatorStatus
16.56 Функция BioAPI_GetLastErrorInfo
16.57 Функция BioAPI_Cancel
16.58 Функция BioAPI_Free
16.59 Функция BioAPI_RegisterBSP
16.60 Функция BioAPI_UnregisterBSP
16.61 Функция BioAPI_RegisterBFP

16.62 Функция BioAPI_UnregisterBFP .....
17 Функции обратного вызова, определенные в БиоАПИ, и соответствующие сообщения ПМО БиоАПИ .....
17.1 Функция обратного вызова BioAPI_EVENT_HANDLER .....
17.2 Функция обратного вызова BioAPI_GUI_SELECT_EVENT_HANDLER .....
17.3 Функция обратного вызова BioAPI_GUI_STATE_EVENT_HANDLER
17.4 Функция обратного вызова BioAPI_GUI_PROGRESS_EVENT_HANDLER .....
18 Концептуальные таблицы .....
18.1 Концептуальная таблица MasterEndpoints .....
18.2 Концептуальная таблица VisibleEndpoints .....
18.3 Концептуальная таблица VisibleBSPRegistrations .....
18.4 Концептуальная таблица VisibleBFPRegistrations .....
18.5 Концептуальная таблица RunningBSPLocalReferences .....
18.6 Концептуальная таблица RunningBSPRemoteReferences .....
18.7 Концептуальная таблица UnitEventNotifcationDisablers .....
18.8 Концептуальная таблица AttachSessionLocalReferences .....
18.9 Концептуальная таблица AttachSessionRemoteReferences .....
18.10 Концептуальная таблица GUIEventLocalSubscriptions .....
18.11 Концептуальная таблица GUIEventRemoteSubscriptions .....
18.12 Концептуальная таблица GUIEventRedirectors .....
18.13 Концептуальная таблица ApplicationOwnedMemoryBlocks .....
19 Преобразования между переменной указателя Си и соответствующим компонентом ACH.1 (1) .....
20 Преобразования между переменной указателя Си и соответствующим компонентом ACH.1 (2) .....
21 Преобразования между переменной указателя Си и соответствующим компонентом ACH.1 (3) .....

22 Инициализация и проверка переменной указателя Си, не имеющей соответствующего компонента АСН.1 .....
23 Определение главенствующей конечной точки и УУИД продукта ПБУ из УУИД ПБУ.....
24 Определение главенствующей конечной точки и исходного дескриптора ПБУ из локального обработчика ПБУ .....
25 Преобразования УУИД ПБУ.....
26 Преобразования дескрипторов ПБУ .....
27 Обработка входящего вызова функции путем обмена с второстепенной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ.....
28 Обработка входящего сообщения запроса ПМО БиоАПИ путем внутреннего вызова функции БиоАПИ .....
29 Предоставление нулю или более подписчикам информации о модуле операций.....
30 Предоставление подписчику информации об операции выбора ГИП .....
31 Предоставление подписчику информации об операции состояния ГИП ...
32 Предоставление подписчику информации об операции прогресса ГИП....
33 Обработка непреобразуемых значений Си.....
Приложение А (обязательное) Спецификация привязки TCP/IP.....
Приложение В (обязательное) Спецификация обнаружения и объявления в привязке TCP/IP .....
Приложение С (обязательное) Спецификация привязки SOAP/HTTP.....
Приложение D (обязательное) Разъяснение минимальных требований для простых систем .....
Приложение Е (обязательное) Возможные сценарии, включающие использование протокола межсетевого обмена БиоАПИ .....
Приложение F (обязательное) Формальные модули АСН.1.....
Приложение G (справочное) Библиография .....

Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации.....

Введение

Настоящий стандарт устанавливает протокол межсетевого обмена БиоАПИ (ПМО БиоАПИ) и определяет синтаксис, семантику и кодировку ряда сообщений (БиоАПИ-сообщений ПМО), которые позволяют соответствующим приложениям БиоАПИ отправлять запросы биометрических операций БиоАПИ соответствующему поставщику биометрических услуг (ПБУ) с помощью узла либо границ процесса, а также получать информацию о событиях, происходящих в соответствующих удаленных ПБУ. ПМО БиоАПИ также определяет расширение архитектуры и характеристик структуры БиоАПИ, определенных в ИСО/МЭК 19784-1, которые поддерживают создание, обработку, отправление и получение сообщений ПМО БиоАПИ.

Наиболее вероятной областью применения настоящего стандарта являются случаи, когда правительство конкретной страны принимает решение о создании системы биометрической регистрации и аутентификации, предусматривающее создание централизованного хранилища биометрических данных граждан данной страны с возможностью доступа к этому хранилищу с биометрических устройств в учреждениях здравоохранения, социального обеспечения, миграционной службы и силовых ведомств. Это одно из нескольких направлений, в которых возможно применение ПМО БиоАПИ.

ПМО БиоАПИ разработан таким образом, что соответствующие ему реализации не требует полных функциональных возможностей структуры БиоАПИ. Для обеспечения различных степеней поддержки таких функциональных возможностей в настоящем стандарте определено несколько классов соответствия. Это позволяет создавать облегченные реализации, соответствующие настоящему стандарту, в которых поддержка приложений, соответствующих БиоАПИ, или поставщиков биометрических услуг (ПБУ), соответствующих БиоАПИ, невозможна или не требуется.

В настоящем стандарте использована нотация АСН.1 (см. серию рекомендаций ITU-T X.680 и комплекс стандартов ИСО/МЭК 8824-1) для определения сообщений протокола.

В разделах 7 – 11 приведены общие справочные сведения. В разделе 12, и некоторых приложениях - нормативные сведения.

В разделе 7 приведено описание инфраструктуры ПМО БиоАПИ.

В разделе 8 установлен механизм удаления уведомлений о событиях графического интерфейса пользователя.

В разделе 9 приведены некоторые примеры возможных конфигураций системы с использованием ПМО БиоАПИ.

В раздел 10 приведен формат биометрических данных, передаваемых ПМО БиоАПИ.

В разделе 11 приведена идентификация конечных точек ПМО БиоАПИ, приложений и ПБУ.

В разделе 12 приведен краткий обзор обмена сообщениями ПМО БиоАПИ.

В разделе 13 приведены общие положения, относящиеся к другим разделам.

В раздел 14 установлен общий синтаксис ПМО БиоАПИ сообщений.

В разделе 15 приведены соответствия между типами БиоАПИ и соответствующими типами АСН.1, которые являются компонентами сообщений ПМО БиоАПИ.

В разделе 16 установлен синтаксис некоторых индивидуальных ПМО БиоАПИ сообщений и действия, выполняемые при поступлении обратного вызова функции БиоАПИ, либо сообщений ПМО БиоАПИ, относящихся к вызову функции БиоАПИ.

В разделе 17 установлен синтаксис некоторых индивидуальных ПМО БиоАПИ сообщений и действия, которые должны быть выполнены при поступлении вызова функции БиоАПИ или сообщения, относящегося к вызову функции БиоАПИ.

В раздел 18 приведены концептуальные таблицы, которые следует использовать при реализации приложений.

В разделах 19 – 33 приведены специальные положения, относящиеся к другим разделам.

В приложении А определена привязка протокола TCP/IP к ПМО БиоАПИ.

В приложение В установлены дополнительные условия привязки протокола TCP/IP к ПМО БиоАПИ.

В приложении С определена привязка протокола SOAP/HTTP к ПМО БиоАПИ.

В приложении D приведены минимальные требования для простых систем.

В приложении Е приведены примеры сценариев, в которых может использоваться ПМО БиоАПИ.

В приложении F приведена полная АСН.1 спецификация ПМО БиоАПИ.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационные технологии
БИОМЕТРИЯ
Протокол межсетевого обмена БиоАПИ

Information technology. Biometrics. BioAPI Interworking Protocol

Дата введения – 2015 – 01 – 01

1 Область применения

1.1 Настоящий стандарт устанавливает требования к синтаксису, семантике и кодировке ряда сообщений (сообщения ПМО БиоАПИ), которые позволяют соответствующим приложениям БиоАПИ совершать запросы биометрических операций в соответствующем поставщике биометрических услуг (ПБУ) с помощью узла или границы процесса, а также получать информацию о событиях, происходящих в удаленных ПБУ.

Примечание – Локальный и удаленный узлы или процессы могут содержать ПБУ, который обеспечивает хранение и поиск записей биометрической информации, обработку или сравнение таких записей или сбор биометрической информации от одного или более биометрических сканеров. Отдельный узел или процесс могут содержать одновременно одно или более приложений с доступом к удаленному ПБУ и один или более ПБУ, доступных для удаленных приложений.

1.2 В настоящем стандарте также установлены требования к расширению для архитектуры и поведение структуры БиоАПИ, которая регулирует создание, обработку, отправление и получение сообщений ПМО БиоАПИ. Структура БиоАПИ, соответствующая требованиям настоящего стандарта (ПМО БиоАПИ-поддерживающая структура), создает, обрабатывает, отправляет и получает сообщения ПМО БиоАПИ с вызовом и обратным вызовом функции БиоАПИ. Исходящие сообщения ПМО БиоАПИ могут создаваться и отправляться структурой в рамках процесса обработки входящего вызова или обратного вызова. Входящие сообщения ПМО БиоАПИ

Издание официальное

могут вызвать реакцию со стороны структуры, такую же, как при получении вызова или обратного вызова.

1.3 Настоящий стандарт допускает создание, обработку, отправку и получение сообщений ПМО БиоАПИ программмным объектом (далее – объект класса ПМО БиоАПИ), который может необязательно являться поддерживающей структурой ПМО БиоАПИ.

Примечание – Последнее позволяет создавать облегченную реализацию настоящего стандарта, в которой поддержка соответствущих приложений БиоАПИ или соответствущих ПБУ БиоАПИ не обязательна или не требуется. Существенные видимые отличия между сообщениями ПМО БиоАПИ, которые созданы и отправлены объектом класса ПМО БиоАПИ, и сообщениями, которые созданы и отправлены поддерживающей структурой ПМО БиоАПИ, отсутствуют. Тем не менее, если поддерживающая структура ПМО БиоАПИ должна полностью и должным образом осуществлять связь, определенную в настоящем стандарте, между сообщениями ПМО БиоАПИ и вызовом или обратным запросом функции БиоАПИ, для объекта класса ПМО БиоАПИ такое требование не является обязательным (см. раздел 6).

1.4 В настоящем стандарте установлены требования к использованию любого из нескольких общедоступных транспортных протоколов для передачи сообщений ПМО БиоАПИ между двумя объектами программного обеспечения (конечными точками ПМО БиоАПИ).

1.5 Настоящий стандарт не распространяется на стандартизацию блоков биометрических данных, содержащих исходные, промежуточные или обработанные биометрические образцы.

Примечание – Требования к стандартизации таких форматов приведены в стандартах комплекса ИСО/МЭК 19794.

1.6 Настоящий стандарт не распространяется на стандартизацию записей биометрической информации, которые содержат один или несколько блоков биометрических данных вместе с идентификационной и другой вспомогательной информацией.

Примечание – Требования к стандартизации элементов таких форматов приведены в ИСО/МЭК 19785-1, в котором также содержится спецификация некоторых стандартизированных форматов записей биометрических информации.

1.7 Настоящий стандарт не устанавливает алгоритмы сравнения для биометрической идентификации или верификации.

1.8 Настоящий стандарт не устанавливает требования безопасности, однако приведены способы установления связи с безопасными транспортными протоколами с целью поддержания безопасного обмена между конечными точками ПМО БиоАПИ.

1.9 Настоящий стандарт не распространяется на классификацию, определения или требования к функциональным характеристикам биометрических систем.

1.10 Настоящий стандарт определяет версию 1 протокола межсетевого обмена БиоАПИ (ПМО БиоАПИ) и присваивает данному протоколу значение идентификатора объекта ACH.1 (стандарт международной организации по стандартизации 24708 версии (1)) (см. рекомендации ITU-T X.680 | ИСО/МЭК 8824-1).

1.11 Настоящий стандарт предусматривает поддержку версии 2.1 БиоАПИ. В ИСО/МЭК 19784-1 приведены определения версий 2.0 и 2.1 БиоАПИ.

2 Нормативные ссылки

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

Рекомендации ITU-T X.667 (2004)| ИСО/МЭК 9834-8:2005, Информационные технологии. Взаимосвязь открытых систем. Процедуры работы уполномоченных по регистрации ВОС: создание, регистрация

универсальных уникальных идентификаторов (УУИД) и их использование в качестве компонентов идентификатора объекта ACH.1 (ISO/IEC 9834-8:2005, Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components)

Рекомендации ITU-T X.680 (2002) | ИСО/МЭК 8824-1:2002, Информационные технологии. Абстрактная синтаксическая нотация версии один (ACH.1). Спецификация основной нотации (ISO/IEC 8824-1:2002, Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation)

Рекомендации ITU-T X.681 (2002) | ИСО/МЭК 8824-2:2002, Информационные технологии. Абстрактная синтаксическая нотация версии один (ACH.1). Спецификация информационного объекта (ISO/IEC 8824-2:2002, Information technology – Abstract Syntax Notation One (ASN.1): Information object specification)

Рекомендации ITU-T X.682 (2002) | ИСО/МЭК 8824-3:2002, Информационные технологии. Абстрактная синтаксическая нотация версии один (ACH.1). Спецификация ограничения (ISO/IEC 8824-3:2002, Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification)

Рекомендации ITU-T X.683 (2002) | ИСО/МЭК 8824-4:2002, Информационные технологии. Абстрактная синтаксическая нотация версии один (ACH.1). Параметризация спецификации ACH.1 (ISO/IEC 8824-4:2002, Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications)

Рекомендации ITU-T X.691 (2002) | ИСО/МЭК 8825-2:2002, Информационные технологии. Правила кодирования ACH.1. Спецификация правил уплотненного кодирования (PER). (ISO/IEC 8825-2:2002, Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER))

Рекомендации ITU-T X.693 (2001) | ИСО/МЭК 8825-4:2002,

Информационные технологии. Правила кодирования АСН.1. Правила кодирования XML (XER) (ISO/IEC 8825-4:2002, Information technology – ASN.1 encoding rules: XML Encoding Rules (XER))

Рекомендации ITU-T X.693 (2001)/Доп.1 (2003) | ИСО/МЭК 8825-4:2002/Доп.1:2004, Информационные технологии. Правила кодирования АСН.1. Правила кодирования XML (XER). Дополнение 1: EXTENDED-XER (ISO/IEC 8825-4:2002/Amd.1:2004, Information technology – ASN.1 encoding rules: XML Encoding Rules (XER) – Amendment 1: XER encoding instructions and EXTENDED-XER)

ИСО/МЭК ТО 8802-1:2001, Информационные технологии. Телекоммуникации и информационный обмен между системами. Локальные и общегородские сети. Специальные требования. Часть 1. Обзор стандартов на локальные сети (ISO/IEC TR 8802-1:2001, Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 1 : Overview of Local Area Network Standards)

ИСО/МЭК 19784-1:2006, Информационные технологии. Биометрический программный интерфейс. Часть 1. Спецификация биометрического программного интерфейса (ISO/IEC 19784-1:2006, Information technology – Biometric application programming interface – Part 1: BioAPI specification)

ИСО/МЭК 19785-1:2006, Информационные технологии. Единая структура форматов обмена биометрическими данными (ЕСФОБД). Часть 1. Спецификация элементов данных (ISO/IEC 19785-1:2006, Information technology – Common Biometric Exchange Formats Framework – Part 1: Data element specification)

ИСО/МЭК 19785-3:2007 Информационные технологии. Единая структура форматов обмена биометрическими данными (ЕСФОБД). Часть 3. Спецификации форматов ведущей организации (ISO/IEC 19785-3:2007, Information technology – Common Biometric Exchange Formats Framework – Part

3: Patron format specifications)

ИСО/МЭК 19794 (все части), Информационные технологии. Форматы обмена биометрическими данными (ISO/IEC 19794 (all parts), Information technology – Biometric data interchange formats)

IETF RFC 768 (1980), Протокол датаграмм клиента (IETF RFC 768 (1980), User Datagram Protocol)

IETF RFC 791 (1981), Межсетевой протокол (IP) (IETF RFC 791 (1981), Internet Protocol)

IETF RFC 793 (1981), Протокол управления передачей (TCP) (IETF RFC 793 (1981), Transmission Control Protocol)

IETF RFC 826 (1982), Протокол преобразования адресов Ethernet (ARP) (IETF RFC 826 (1982), Ethernet Address Resolution Protocol)

IETF RFC 1945 (1996), Протокол передачи гипертекста HTTP/1.0 (IETF RFC 1945 (1996), Hypertext Transfer Protocol – HTTP/1.0)

IETF RFC 2131 (1997), Протокол динамической конфигурации узла для протокола IPv4 (DHCPv4) (IETF RFC 2131 (1997), Dynamic Host Configuration Protocol)

IETF RFC 2136 (1997), Динамические обновления системы доменных имен (Обновление DNS) (IETF RFC 2136 (1997), Dynamic Updates in the Domain Name System (DNS UPDATE))

IETF RFC 2462 (1998), Динамическое назначение адресов в протоколе IPv6 IETF RFC 2462 (1998), IPv6 Stateless Address Autoconfiguration)

IETF RFC 2616 (1999), Протокол передачи гипертекста HTTP/1.1 (IETF RFC 2616 (1999), Hypertext Transfer Protocol – HTTP/1.1)

IETF RFC 2818 (2000), Применение протокола HTTP над TLS (IETF RFC 2818 (2000), HTTP Over TLS)

IETF RFC 3315 (2003), Протокол динамической конфигурации узла для протокола IPv6 (DHCPv6) (IETF RFC 3315 (2003), Dynamic Host Configuration Protocol for IPv6 (DHCPv6))

IETF RFC 3927 (2005), Динамическая конфигурация Link-Local адресов IPv4 (IETF RFC 3927 (2005), Dynamic Configuration of IPv4 Link-Local Addresses)

IETF RFC 3987 (2005), Интернационализированный идентификатор ресурса (ИИР) (IETF RFC 3987 (2005), Internationalized Resource Identifiers (IRIs))

IETF RFC 4443 (2006), Межсетевой протокол управляющих сообщений (ICMPv6) для межсетевого протокола версии 6 (IPv6) (IETF RFC 4443 (2006), Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification)

W3C SOAP 1.2:2007, Простой протокол доступа к объектам, версия 1.2 (W3C SOAP 1.2:2007, SOAP Version 1.2)

W3C SOAP MTOM:2005, Механизм оптимизации передачи сообщения протокола SOAP (W3C SOAP MTOM:2005, SOAP Message Transmission Optimization Mechanism)

W3C XMLENC:2002, Процесс шифрования данных и представления результата в XML (W3C XMLENC:2002, XML Encryption Syntax and Processing)

W3C XMLDSIG:2002, XML – Синтаксис представления цифровых подписей и правила их обработки для XML.(W3C XMLDSIG:2002, XML – Signature Syntax and Processing)

3 Термины и определения

В настоящем стандарте применены термины, определенные в ИСО/МЭК 19784-1 и ИСО/МЭК 19785-1, а также следующие термины и соответствующие определения:

3.1 **сообщение подтверждения ПМО БиоАПИ** (acknowledgement BIP message): сообщение ПМО БиоАПИ, передающееся в ответ (подтверждение) на соответствующее сообщение уведомления ПМО БиоАПИ.

Примечание – Не все сообщения уведомления ПМО БиоАПИ предполагают соответствующие сообщения подтверждения ПМО БиоАПИ.

3.2 **поддерживающая инфраструктура ПМО БиоАПИ** (BIP-enabled framework): Расширенная версия инфраструктуры БиоАПИ, способная к созданию, обработке, отправлению и получению сообщений ПМО БиоАПИ в тесной связи с запросами и обратными запросами функции БиоАПИ.

Примечание – Не все реализации поддерживающей инфраструктуры ПМО БиоАПИ являются инфраструктурами БиоАПИ согласно ИСО/МЭК 19784-1 (см. примечание к 6.7).

3.3 **конечная точка ПМО БиоАПИ** (BIP endpoint): Объект, идентифицированный ИИР конечной точки, который способен к отправлению и получению сообщений ПМО БиоАПИ и содержит либо активный объект класса ПМО БиоАПИ, либо активную поддерживающую инфраструктуру ПМО БиоАПИ с одним компонентом реестра, без или с одним активным приложением БиоАПИ и без или с одним или более активным ПБУ.

3.4 **связь ПМО БиоАПИ** (BIP link): Логическая связь между двумя конечными точками ПМО БиоАПИ, в которой обязательно присутствует канал связи запрос/ответ и необязательно – канал связи уведомление/подтверждение, а также одна конечная точка ПМО БиоАПИ играет роль главной, а другая – второстепенной.

Примечание – Между любой парой конечных точек ПМО БиоАПИ должно существовать не менее одной связи ПМО БиоАПИ при любом распределении ролей. Если в связи ПМО БиоАПИ присутствуют два канала связи, то одна из двух конечных точек ПМО БиоАПИ будет главной конечной точкой в обоих каналах связи, в то время как вторая будет второстепенной конечной точкой в тех же двух каналах связи. Связь ПМО БиоАПИ не может существовать между парой конечных точек ПМО БиоАПИ, если обе точки будут главными или второстепенными.

3.5 **сообщение ПМО БиоАПИ** (BIP message): Сообщение, которое можно отправить от конечной точки ПМО БиоАПИ до другой конечной точки ПМО БиоАПИ с помощью канала связи.

3.6 **УУИД доступа ПБУ** (BSP access UUID): динамически создаваемый и назначаемый поддерживающей ПМО БиоАПИ инфраструктурой для ПБУ непостоянный УУИД, доступный для загрузки в заданную конечную точку

ПМО БиоАПИ (локальную или удаленную) и однозначно идентифицирующий ПБУ и, неявно, конечную точку ПМО БиоАПИ.

Примечание – УУИД доступа ПБУ лишен смысла вне конечной точки ПМО БиоАПИ, в которой он был создан.

3.7 **УУИД продукта ПБУ** (BSP product UUID): постоянный УУИД, определенный поставщиком программного обеспечения ПБУ.

Примечание1 – После многократной переустановки ПБУ на различных системах предполагается сохранение УУИД продукта ПБУ.

Примечание 2 – УУИД продукта ПБУ может быть создан и зарегистрирован на сайте http://www.itu.int/ITU-T/asn1/UUID.html (см. ITU-T Rec. X.667).

3.8 **ЕСФОБД** (CBEFF): Элементы информации и формат записи биометрической информации, определенные в ИСО/МЭК 19785-1.

3.9 **ИИР конечной точки** (endpoint IRI): ИИР, однозначно идентифицирующий конечную точку ПМО БиоАПИ.

Примечание – Ограничения формы данного ИИР отсутствуют. В общем случае (если иное не предписано правилами обязательной спецификации), зависимость между ИИР схемы ИИР конечной точки и привязкой(ами), которая(ые) поддерживаются основной реализацией, отсутствует. Таким образом, ИИР конечной точки может не предоставлять информацию достаточную для того, чтобы определить местонахождение конечной точки ПМО БиоАПИ в сети.

3.10 **объект класса ПМО БиоАПИ** (generic BIP entity): Объект программного обеспечения, который способен создавать, обрабатывать, отправлять и получать сообщения ПМО БиоАПИ, но не обязательно реализует программный интерфейс приложений БиоАПИ и полную функциональность инфраструктуры БиоАПИ (включая доступ к реестру компонентов и локальному ПБУ).

3.11 **канал связи** (link channel): Логическая связь между двумя конечными точками ПМО БиоАПИ путем привязки транспортного протокола, в которой одна конечная точка ПМО БиоАПИ играет роль главной, а другая – второстепенной.

3.12 **главная конечная точка (заданная конечная точка ПМО БиоАПИ)** (master endpoint (of a given BIP endpoint)): Конечная точка ПМО БиоАПИ, которая логически связана путем связи ПМО БиоАПИ с второстепенной конечной точкой ПМО БиоАПИ и способна отправлять сообщения запроса ПМО БиоАПИ к заданной конечной точке ПМО БиоАПИ и обрабатывать полученные сообщения уведомления ПМО БиоАПИ.

Примечание – Заданная конечная точка ПМО БиоАПИ может одновременно играть как главную роль, так и второстепенную роль с любым числом других конечных точек ПМО БиоАПИ, при условии сохранения правильного набора связей ПМО БиоАПИ сохранен.

3.13 **сообщение уведомления ПМО БиоАПИ** (notification BIP message): Сообщение уведомления ПМО БиоАПИ об операции «заинтересованности», передаваемое к принимающей конечной точке ПМО БиоАПИ.

3.14 **канал связи уведомление/подтверждение** (notification/ acknowledgement link channel): Канал связи, по которому передаются сообщения уведомления и подтверждения ПМО БиоАПИ.

Примечание – Канал связи уведомления/подтверждения может существовать только как часть связи ПМО БиоАПИ. Его присутствие в связи ПМО БиоАПИ является дополнительным.

3.15 **политика удаленного доступа** (remote access policy): Политика, определяющая для каких локального ПБУ, локального ПБФ, и блоков БиоАПИ (управляемых указанными ПБУ и ПБФ) конечная точка ПМО БиоАПИ будет доступной для использования другой конечной точкой ПМО БиоАПИ.

3.16 **сообщение запроса ПМО БиоАПИ** (request BIP message): Сообщение ПМО БиоАПИ, передающее запрос о выполнении действия принимающей конечной точкой ПМО БиоАПИ.

3.17 **канал связи запроса/ответа** (request/response link channel): Канал связи, постоянно присутствующий в связи ПМО БиоАПИ, через который передаются сообщения запроса и ответа ПМО БиоАПИ (см. 3.4).

Примечание – Канал связи запроса/ответа может существовать только как часть связи ПМО БиоАПИ.

3.18 **сообщение ответа ПМО БиоАПИ** (response BIP message): Сообщение ПМО БиоАПИ, передающее ответ на предшествующее сообщение запроса ПМО БиоАПИ.

3.19 **второстепенная конечная точка (заданная конечная точка ПМО БиоАПИ)** (slave endpoint (of a given BIP endpoint): конечная точка ПМО БиоАПИ, с которой логически связана с помощью связи ПМО БиоАПИ заданная (главная) конечная точка ПМО БиоАПИ и которая способна к обработке сообщений запроса ПМО БиоАПИ, полученных от заданной конечной точки ПМО БиоАПИ, и отправке к ней сообщений уведомления ПМО БиоАПИ.

Примечание - Заданная конечная точка ПМО БиоАПИ может одновременно играть как главную так и второстепенную роль с любым числом других конечных точек ПМО БиоАПИ, при условии, сохранения правильного набора связей ПМО БиоАПИ.

3.20 **привязка транспортного протокола** (transport protocol binding): Физическая реализация канала связи, при которой определяется, какой транспортный протокол будет использоваться, как будут кодироваться сообщения ПМО БиоАПИ, каким образом будет составлено сообщение транспортного уровня, несущее закодированные сообщения ПМО БиоАПИ, и другие параметры использования транспортного протокола.

4 Сокращения

В настоящем стандарте применены следующие сокращения:

ПОА (ARP) – протокол определения адреса (address resolution protocol);

АСН.1 (ASN.1) – абстрактная синтаксическая нотация, версия 1 (abstract syntax notation one);

ББД (BDB) – блок биометрических данных (biometric data block);

ПБФ (BFP) – поставщик биометрической функции (biometric function providers);

ПМО БиоАПИ (BIP) – протокол межсетевого обмена БиоАПИ (BioAPI interworking protocol);

ЗБИ (BIR) – запись биометрической информации (biometric information record);

ПБУ (BSP) – поставщик биометрической услуги (biometric service providers);
DHCP – протокол динамической конфигурации узла (dynamic host configuration protocol);
ИПФ (FPI) – интерфейс поставщика функции (function provider interface);
HTTP – протокол передачи гипертекста (hypertext transfer protocol);
HTTPS – протокол защищенной передачи гипертекста (HTTP over secure socket layer);
IP – межсетевой протокол (см. IETF RFC 791) (internet protocol);
ИИР (IRI) – интернационализированный идентификатор ресурса (internationalized resource identifier);
УДК (MAC) – управление доступом к среде (media access control);
МТОМ – механизм оптимизации передачи сообщений (message transmission optimization mechanism);
ПСК (PER) – правила сжатого кодирования (packed encoding rules);
SOAP – простой протокол доступа к объектам (simple object access protocol);
ИПУ (SPI) – интерфейс поставщика услуги (service provider interface);
TCP – протокол управления передачей (transmission control protocol);
UDP – протокол пользовательских датаграмм (user datagram protocol);
УУИД (UUID) – универсальный уникальный идентификатор (universally unique identifier).

5 Условные обозначения

В настоящем стандарте для облегчения восприятия текста применяются следующие выделенные полужирным шрифтом и путем подчеркивания условные обозначения, при этом для понимания стандарта они не являются ключевыми:

- языковые имена и определения Си (типы определения БиоАПИ, определения функций БиоАПИ и входные параметры функций БиоАПИ) выделены следующим образом: **BioAPI_Init;** исходные параметры функций БиоАПИ выделены следующим образом: **Response;**

- языковые имена и определения АСН.1 (такие как типы определений АСН.1, имена компонентов АСН.1 и типы сообщений ПМО БиоАПИ), выделены следующим образом **masterEndpointIRI;**

- имена и документы XML (такие как определения схемы XML, глобальные имена элементов, и примеры закодированных сообщений XML ПМО БиоАПИ) выделены следующим образом: **<bip:VerifyMatch/>.**

6 Соответствие

6.1 В разделах 7 - 33 устанавлины требования к синтаксису и семантике сообщений ПМО БиоАПИ, а также к поведению поддерживающей инфраструктуры ПМО БиоАПИ при получении входящих запросов функции БиоАПИ (совершенных локальным приложением), обратных запросов (совершенных локальным ПБУ) и сообщений ПМО БиоАПИ.

6.2 Поддерживающая инфраструктура ПМО БиоАПИ является одним из типов реализации настоящего стандарта. Другим типом реализации является объект класса ПМО БиоАПИ, который способен создавать, обрабатывать, отправлять и получать сообщения ПМО БиоАПИ, при этом который не обязательно реализует БиоАПИ ПИП и полную внутреннюю функциональность инфраструктуры БиоАПИ (включая доступ к реестру компонентов и к локальному ПБУ).

6.3 Требования к соответствию согласно настоящему стандарту должны иметь два уровня (уровень 1 и уровень 2) и три класса ролей (главная роль, второстепенная роль и двойная роль), что, в общем, обеспечивает шесть классов соответствия, представленных согласно таблице 1:

Таблица 1 - Классы соответствия

		Уровень соответствия	
		1 (объект класса ПМО БиоАПИ)	2 (поддерживающая инфраструктура ПМО БиоАПИ)
Класс роли	Главная роль	Объект класса ПМО БиоАПИ с главной ролью	Поддерживающая инфраструктура ПМО БиоАПИ с главной ролью
	Второстепенная роль	Объект класса ПМО БиоАПИ с второстепенной ролью	Поддерживающая инфраструктура ПМО БиоАПИ с второстепенной ролью
	Двойная роль	Объект класса ПМО БиоАПИ с двойной ролью	Поддерживающая инфраструктура ПМО БиоАПИ с двойной ролью

6.4 Положения разделов 7 – 33 применяют для каждого уровня соответствия следующим образом:

a) для поддерживающей инфраструктуры ПМО БиоАПИ положения следует применять в соответствии с приведенными указаниями;

b) для объекта класса ПМО БиоАПИ положения следует интерпретировать, как относящиеся к идеальной инфраструктуре, концептуально представленной в рамках объекта класса ПМО БиоАПИ, без предположения, что идеальный БиоАПИ и ПБУ БиоАПИ идеальной инфраструктуры обозримы для любого внешнего наблюдателя или теста.

Примечание – Внутренняя структура объекта класса ПМО БиоАПИ является не только невидимой, но также полностью недоступной и не определяемой для соответствия или тестирования на соответствие.

6.5 Объект программного обеспечения может быть определен как поддерживающая инфраструктура ПМО БиоАПИ с главной ролью, только в том случае если он:

a) предоставляет локальному приложению доступ к программному интерфейсу приложения БиоАПИ;

b) обрабатывает поступающие запросы БиоАПИ от локального приложения, согласно в разделам 7 - 33;

c) обрабатывает поступающее сообщения уведомления ПМО БиоАПИ в случаях, указанных в соответствующих разделах и

d) никогда не производит сообщения ПМО БиоАПИ, кроме случаев, определенных в соответствующих разделах.

6.6 Объект программного обеспечения может быть определен, как поддерживающая инфраструктура ПМО БиоАПИ с второстепенной ролью, только в том случае, если он:

a) использует программный интерфейс приложений БиоАПИ, представленный локальным поддерживающим БиоАПИ ПБУ, для взаимодействия с ними;

b) обрабатывает поступающие обратные вызовы от локального ПБУ, согласно разделам 7 – 33;

c) обрабатывает поступающие сообщения ПМО БиоАПИ запроса в случаях, указанных в соответствующих разделах и

d) никогда не производит сообщения ПМО БиоАПИ кроме случаев, указанных в соответствующих разделах.

6.7 Объект программного обеспечения может быть определен, как поддерживающая инфраструктура ПМО БиоАПИ с двойной ролью, только в том случае, если он удовлетворяет требованиям поддерживающей инфраструктуры ПМО БиоАПИ с главной и второстепенной ролью.

Примечание – Поддерживающая инфраструктура ПМО БиоАПИ с двойной ролью – единственный тип реализации ПМО БиоАПИ, который должен поддерживать как программный интерфейс приложения БиоАПИ, так и использование локального ПБУ, в связи с чем этот тип реализации ПМО БиоАПИ считается единственным, который можно считать в соответствии с требованиями ИСО/МЭК 19784-1, истинным супернабором инфраструктуры БиоАПИ.

6.8 Объект программного обеспечения может быть определен как объект класса ПМО БиоАПИ (с главной ролью, второстепенной ролью или двойной ролью), только в том случае, если для любой возможной последовательности (произвольной длины) входящих и исходящих сообщений ПМО БиоАПИ существует возможность определения соответствующей последовательности действий, которые могут быть выполнены концептуально

представленной в объекте программного обеспечения идеальной поддерживающей инфраструктурой ПМО БиоАПИ (с соответствующей ролью), и результатом которых может быть указанная последовательность сообщений ПМО БиоАПИ.

Примечание – В разделах 6 - 32 установлены требования, гарантирующие, что при обмене сообщениями ПМО БиоАПИ одной реализации не обязательно надо знать класс соответствия другой. Так как другая реализация ПМО БиоАПИ будет либо соответствующей поддерживающей инфраструктурой ПМО БиоАПИ, либо (если это объект класса ПМО БиоАПИ) будет вести себя, как будто он содержит поддерживающую инфраструктуру ПМО БиоАПИ. Например, невозможно зная только о сообщениях ПМО БиоАПИ, поступающих от другой реализации, определить действительно ли другая реализация использует поддерживающий БиоАПИ ПБУ для выполнения биометрических операций или использует нестандартную внутреннюю архитектуру.

6.9 В соответствии предыдущих подразделов представленные интерфейсы являются объектами тестирования на соответствие по каждому классу соответствия:

Таблица 2 – Тестирование на соответствие

Класс соответствия	БиоАПИ ПИП	БиоППИ* ПИП	Интерфейс передачи
Объект класса ПМО БиоАПИ с главной ролью			X
Объект класса ПМО БиоАПИ с второстепенной ролью			X
Объект класса ПМО БиоАПИ с двойной ролью			X
Поддерживающая инфраструктура ПМО БиоАПИ с главной ролью	X		X
Поддерживающая инфраструктура ПМО БиоАПИ с второстепенной ролью		X	X

Окончание таблицы 2

Класс соответствия	БиоАПИ ПИП	БиоППИ* ПИП	Интерфейс передачи
Поддерживающая инфраструктура ПМО БиоАПИ с двойной ролью	X	X	X

* - Последовательный периферийный интерфейс.

6.10 Требования к соответствию указанной в приложениях А и В привязки транспортного протокола приведены в каждом приложении. Однако тестирование реализации на соответствие не может быть выполнено, если нет, по крайней мере, хотя бы одной привязки, поддерживаемой как реализацией, так и инструментом тестирования. Кроме того, невозможно оценить соответствие в отношении правильной обработки сообщений уведомления ПМО БиоАПИ, если привязка, используемая в тесте, не поддерживает передачу сообщений уведомления ПМО БиоАПИ.

6.11 Реализация, заявленная на соответствие требованиям приложения А, не должна также соответствовать требованиям приложения В, в то время как реализация, заявленная на соответствие приложению В требует также соответствия приложению А.

6.12 Соответствующие реализации дополнительно должны обнаруживать и обрабатывать битовые комбинации в поступающих сообщениях, которые не являются частью протокола ПМО БиоАПИ (или, которые представляют собой правильные сообщения, которые не были разрешены в предыдущих обменах), способом, обеспечивающим предотвращение отказа сервиса от внешних источников или от «спрятанного» в другой соответствующей системе «троянского коня». Это требование относится ко всем реализациям, которые должны соответствовать требованиям настоящего стандарта, даже если реализация предназначена для использования в физически защищенной сети. Если соответствие заявлено, требование настоящего подраздела не является опциональным.

Примечание – Это требование по существу означает, что соответствующий приемник не делает предположений о правильности кодирования и правильности сообщений, принимаемых даже по связи, установленной от доверенного отправителя ПМО БиоАПИ. Заявленные на соответствие реализации не должны «падать» из-за, например, проблем перегруженного буфера. Ожидается, что, если ПМО БиоАПИ получит широкое применение, будут проведены расширенные испытания, целью которых будет определение неправильных реализаций, которые приводят к незащищенности от отказа сервиса, возникающих как результат неверных сообщений ПМО БиоАПИ, поступающих как от доверенного, так и от сомнительного отправителя.

7 Архитектура ПМО БиоАПИ

В дополнение к понятиям БиоАПИ, таким как ПБУ и локальное приложение, в основе ПМО БиоАПИ лежат понятия поддерживающей инфраструктуры ПМО БиоАПИ, сообщения ПМО БиоАПИ, конечной точки ПМО БиоАПИ, связи ПМО БиоАПИ, главной/второстепенной конечных точек и привязки транспортного протокола.

7.1 Поддерживающие инфраструктуры ПМО БиоАПИ

7.1.1 Одна из целей ПМО БиоАПИ заключается в том, чтобы позволить БиоАПИ приложению использовать удаленный ПБУ теми же способами, как оно может использовать локальный ПБУ.

7.1.2 Различия локального и удаленного ПБУ основано на различиях локальной и удаленной инфраструктур БиоАПИ, как показано далее. Для заданного исполняющегося приложения БиоАПИ локальная инфраструктура БиоАПИ – это исполняемый экземпляр продукта инфраструктуры БиоАПИ, который предоставляет собственный БиоАПИ ПИП приложению, как удаленная инфраструктура БиоАПИ – любой другой исполняемый экземпляр продукта инфраструктуры БиоАПИ (которая может быть расположена либо в компьютере, либо в другом процессе в пределах того же самого компьютера). Локальный ПБУ – загружаемый либо исполняющийся ПБУ, схема которого присутствует в реестре компонентов локальной инфраструктуры БиоАПИ, который вследствие этого доступен для приложения путем БиоАПИ ПИП

заданной локальной инфраструктуры БиоАПИ. Удаленный ПБУ - загружаемый или исполняющийся ПБУ, схема которого присутствует в реестре компонентов удаленной инфраструктуры БиоАПИ, но не в реестре компонентов любой локальной инфраструктуры БиоАПИ, который вследствие этого является недоступным для приложения через БиоАПИ ПИП любой локальной инфраструктуры БиоАПИ.

7.1.3 Существует множество ситуаций, в которых приложению БиоАПИ может потребоваться доступ к удаленному ПБУ. Например, если:

- оборудование сканера физически подключено к одному компьютеру, а приложение исполняется на другом компьютере, при этом оборудование сканера управляется ПБУ, установленном на компьютере, к которому подключено оборудование сканера;
- база шаблонов находится на одном компьютере, а приложение исполняется на другом компьютере, при этом база шаблонов управляется ПБУ, установленном на компьютере, где такая база расположена;
- множество зарегистрированных копий продукта ПБУ, осуществляющих алгоритм сравнения с целью оптимизации использования вычислительных ресурсов становятся доступными для использования на нескольких серверах;
- сервис поиска по списку предоставляется путем использования исполняющегося на сервере ПБУ;
- на компьютере принята ограничивающая доступ к базе шаблонов политика безопасности, при которой ПБУ, управляющему базой шаблонов, и обращающемуся к такому ПБУ приложению должны быть присвоены различные уровни ограничения доступа, в связи с чем они не могут исполняться в одном и том же процессе даже Если их запуск разрешен на одном и том же компьютере;

- исполнение приложения в процессе, отдельном от того, в котором запущен ПБУ, может увеличить надежность всей системы (например, когда приложение «падает», ПБУ может остаться доступным, и наоборот);
- исполнение приложения в процессе, отдельном от того, в котором запущен ПБУ, позволяет множеству параллельных приложений разделить один и тот же исполняемый экземпляр ПБУ, что, в свою очередь, позволяет ПБУ оптимизировать внутреннее управление своими ресурсами.

7.1.4 Доступ к удаленному ПБУ через БиоАПИ ПИП локальной инфраструктуры не поддерживается БиоАПИ, но поддерживается ПМО БиоАПИ.

7.1.5 БиоАПИ, представленный ПМО БиоАПИ-поддерживающей инфраструктурой, синтаксически идентичен тому, который представляется инфраструктурой БиоАПИ (подписи всех функций должны быть одинаковыми), но семантика большинства функций БиоАПИ должна быть расширенной и специализированной. Когда приложение использует БиоАПИ поддерживающей инфраструктуры ПМО БиоАПИ, те же самые последовательности запросов и обратных запросов БиоАПИ работают как для локальных, так и для удаленных ПБУ. Сохраняется возможность замены инфраструктуры БиоАПИ ПМО поддерживающей инфраструктурой БиоАПИ без видимых изменений в поведении (предполагая, что все зарегистрированные ПБУ должны быть повторно зарегистрированы в новой инфраструктуре).

7.1.6 Вышесказанное подразумевает, что нет никакого фундаментального различия между приложением БиоАПИ(не осведомленном о ПМО БиоАПИ), которое использует только локальные ПБУ, и приложением (осведомленным о ПМО БиоАПИ), которое использует: и локальные и удаленные ПБУ, что однако не относится к той части приложения, специфика работы которой связана с локализацией ПБУ (если такие имеются). И локальные и удаленные ПБУ перечислены в массиве, возвращенном **BioAPI_EnumBSPs**; и локальные и удаленные ПБУ «должны быть установлены» путем вызова

BioAPI_BSPLoad и «присоединяются» путем вызова **BioAPI_BSPAttach**; и локальные и удаленные ПБУ могут отправлять приложению уведомления о событиях модуля и уведомления о событиях графического интерфейса пользователя, и т. д. Приложение БиоАПИ будет работать даже в том случае, если инфраструктура БиоАПИ будет заменена ПМО поддерживающей инфраструктурой БиоАПИ, а некоторые или все локальные ПБУ перемещены в удаленную поддерживающую инфраструктуру ПМО БиоАПИ (став, таким образом, удаленными ПБУ).

7.2 Сообщения ПМО БиоАПИ

7.2.1 В настоящем стандарте установлены сообщения, которыми могут обмениваться между собой две поддерживающие инфраструктуры ПМО БиоАПИ (использующие привязку транспортного протокола) и точные правила, обеспечивающие создание и обработку таких сообщений в ответ на запросы и обратные запросы функции БиоАПИ. Поддерживающая инфраструктура ПМО БиоАПИ может создавать, обрабатывать, отправлять и получать сообщения ПМО БиоАПИ. Логическую связь между двумя активными поддерживающими инфраструктурами ПМО БиоАПИ, которая обеспечивает обмен сообщениями ПМО БиоАПИ, называют связью ПМО БиоАПИ.

7.2.2 Протокол передачи сообщений ПМО БиоАПИ смоделирован на БиоАПИ ПИП и тесно с ним связан. Большинство функций и обратных вызовов БиоАПИ обладают соответствующей двумя сообщениями ПМО БиоАПИ (сообщение запроса и сообщение ответа или сообщение уведомления и сообщение подтверждения), но существуют также и исключения.

7.2.3 Абстрактный синтаксис сообщения ПМО БиоАПИ определен в нотации АСН.1. На высшем уровне существует выбор между четырьмя основными видами сообщений (запрос, ответ, уведомление и подтверждение). Протокол передачи сообщений ПМО БиоАПИ не подразумевает использования какого-либо определенного набора правил кодирования для определений типа АСН.1, т. к. передача сообщений ПМО БиоАПИ смоделирована как

концептуальная передача абстрактных значений (см. 7.4). Правила кодирования определяются в индивидуальной привязке транспортного протокола (см. 7.5).

7.2.4 Все четыре вида сообщений ПМО БиоАПИ содержат число связи, которое идентифицирует сообщения, обмен которыми происходит с помощью связи ПМО БиоАПИ (см. 7.4), и идентификатор (положительное целое число), который является либо идентификатором запроса (для запросов и ответов) либо идентификатором уведомления (для уведомлений и подтверждений). Сообщения ПМО БиоАПИ запроса и уведомления, отправляемые с помощью каждой связи ПМО БиоАПИ, имеют независимую нумерацию, начиная с произвольно выбранного числа с последующим увеличением номера при каждом отправлении нового сообщения. Когда идентификатор достигает предельного значения допустимого диапазона (4294967295), он перезапускается с нуля. Сообщение ответа ПМО БиоАПИ должно содержать то же самое число связи и идентификатор, как и соответствующее сообщение запроса ПМО БиоАПИ. Смысл числа связи и идентификатора в ПМО БиоАПИ заключается в том, чтобы облегчить соединение ответов с соответствующими запросами и соединение подтверждений с соответствующими уведомлениями. По идентификатору нельзя определить порядок передачи сообщений ПМО БиоАПИ.

7.3 Конечные точки ПМО БиоАПИ

7.3.1 Несмотря на то, что существует возможность описания обмена сообщениями ПМО БиоАПИ в терминах локальных и удаленных поддерживающих инфраструктур ПМО БиоАПИ, взаимодействующих путем связи ПМО БиоАПИ, в настоящем стандарте рассмотрена иная концепция, основанная на понятии конечной точки ПМО БиоАПИ, которая включает в себя как реализации поддерживающей инфраструктуры ПМО БиоАПИ, так и реализации, которые не являются инфраструктурами, согласно описанию, приведенному в настоящем подразделе.

7.3.2 Конечная точка ПМО БиоАПИ является концептуальным исполняемым объектом программного обеспечения, который идентифицирован уникальным ИИР, и обеспечивает создание, обработку, отправку и получение сообщений ПМО БиоАПИ. Конечная точка ПМО БиоАПИ посылает сообщения ПМО БиоАПИ в другую конечную точку ПМО БиоАПИ с помощью связи ПМО БиоАПИ, которая является логической связью, установленной между двумя конечными точками ПМО БиоАПИ. Одна из двух конечных точек ПМО БиоАПИ в связи ПМО БиоАПИ играет роль главной конечной точки для такой связи, вторая – роль второстепенной конечной точки.

7.3.3 Конечная точка ПМО БиоАПИ может иметь внутреннюю структуру, состоящую из следующих исполняемых компонентов:

a) один исполняемый экземпляр поддерживающей инфраструктуры ПМО БиоАПИ, отвечающий за отправление и получение сообщений ПМО БиоАПИ, которые оказываются отправленными и полученными конечной точкой ПМО БиоАПИ;

b) однокомпонентный реестр БиоАПИ,управляемый инфраструктурой, указанной в перечислении a);

c) ноль или один исполняемый экземпляр приложения БиоАПИ, которое использует БиоАПИ ПИП инфраструктуры, указанной в перечислении a);

d) ноль или один исполняемый экземпляр ПБУ, у которого БиоАПИ ПИП используется инфраструктурой, указанной в перечислении a) и

e) ноль или один исполняемый экземпляр ПБФ, у которого БиоАПИ ИПФ используется некоторыми из БПУ, указанными в перечислении d).

7.3.4 С другой стороны, конечная точка ПМО БиоАПИ может иметь другую конкретную или неопределенную внутреннюю структуру, содержащую объект класса ПМО БиоАПИ, отличающийся от поддерживающей инфраструктуры ПМО БиоАПИ класс соответствия. Однако две конечные точки ПМО БиоАПИ в отношении ПМО БиоАПИ не имеют информации относительно внутренней структуры друг друга. Все положения настоящего

стандарта об обмене сообщениями ПМО БиоАПИ выражены в терминах структуры, посылающей сообщение конечной точке ПМО БиоАПИ (главной и второстепенной) или получающей сообщение от конечной точки ПМО БиоАПИ (главной и второстепенной), выражены без использования терминов поддерживающей инфраструктуры ПМО БиоАПИ, содержащейся в такой конечной точке ПМО БиоАПИ, которая в зависимости от ее класса соответствия может содержать или не содержать поддерживающую инфраструктуру ПМО БиоАПИ.

7.3.5 Существует множество случаев, когда конструктор предпочитает создавать объект класса ПМО БиоАПИ вместо поддерживающей инфраструктуры ПМО БиоАПИ, например следующие:

- вычислительные платформы с ограниченными ресурсами (например, вложенные системы), которые сконструированы для выполнения очень специфического набора биометрических функций и не предназначены для поддержания произвольно выбранных ПБУ или приложений, но которые должны поддерживать обмен сообщениями ПМО БиоАПИ.

- реализации ПМО БиоАПИ, которые предназначены для предоставления ПИП, отличающегося от стандартного БиоАПИ 2.0, определенного в ИСО/МЭК 19784-1, либо для использования модулей биометрической услуги, отличающихся от стандартного БиоАПИ 2.0 ПБУ;

- реализации ПМО БиоАПИ, которые предназначены для поддерживания тех же функциональных возможности инфраструктуры БиоАПИ, однако предоставляют ПИП на языке программирования отличающегося от Си;

- модуль доступа, шлюз, брандмауэры, или другие посредники, которые могут выполнять множество функций (включая трансляцию между различными привязками транспортного протокола, маршрутизацию сообщений, регистрацию сообщений, проверку безопасности, балансировку нагрузки и т. д.); такие функции обычно приводят к созданию другого сообщения ПМО

БиоАПИ того же типа, отправляемого в другую конечную точку ПМО БиоАПИ, вместо биометрической операции, выполняемой данной точкой.

7.3.6 Как было указано выше, внутренняя структура конечной точки ПМО БиоАПИ (Если конечная точка содержит либо поддерживающую инфраструктуру ПМО БиоАПИ, либо вложенную реализацию, нестандартную инфраструктуру, модуль доступа, шлюз и т. д.) «не видна» для любой другой конечной точки ПМО БиоАПИ, которая включена в связь ПМО БиоАПИ с данной конечной точкой ПМО БиоАПИ. Данное условие будет выполняться для любой конечной точки ПМО БиоАПИ при любом обмене сообщениями ПМО БиоАПИ.

7.3.7 Пределы, ограничивающие число приложений БиоАПИ в конечной точке ПМО БиоАПИ (не более одного) и число ИИР конечной точки (не более одного), не предотвращают существование реализаций по настоящему стандарту, в которых реализация поддерживающей инфраструктуры ПМО БиоАПИ одновременно поддерживает множество приложений БиоАПИ и множество ИИР конечной точки. Такая реализация может быть смоделирована, как ряд концептуальных конечных точек ПМО БиоАПИ, каждая из которых содержит только один ИИР конечной точки и не более одного приложения БиоАПИ. Таким образом, вышесказанное относится к каждой из таких концептуальных конечных точек ПМО БиоАПИ.

7.4 Связи ПМО БиоАПИ

7.4.1 Связь ПМО БиоАПИ – это логическая связь, установленная между двумя конечными точками ПМО БиоАПИ для обмена сообщениями ПМО БиоАПИ. Существует множество способов понимания связи ПМО БиоАПИ на физическом уровне.

7.4.2 Термин «отправлять» («send»), использующийся в настоящем стандарте, означает концептуальную передачу сообщения ПМО БиоАПИ от конечной точки ПМО БиоАПИ (отправитель) в связи ПМО БиоАПИ между конечной точкой отправления и конечной точкой получения. Передаваемое

сообщение является абстрактным значением типа **BIPMessage** (см. раздел 14), которое помещается в связь, как результат отправления. Термин «получать» («receive»), использующийся в настоящем стандарте, означает концептуальную передачу сообщения ПМО БиоАПИ от связи ПМО БиоАПИ до конечной точки ПМО БиоАПИ получения. Передаваемое сообщение удаляется из связи после его получения. На этом уровне не рассматривают предположения о физической природе связи ПМО БиоАПИ, об основном транспортном протоколе, о кодировании передаваемого сообщения ПМО БиоАПИ и о временных или связанных с местом аспектах отправления или получения (таких как ожидание и организация очереди).

7.4.3 Роли «главный» и «второстепенный» относительно заданной связи ПМО БиоАПИ ограничивают типы сообщений ПМО БиоАПИ, которые конечная точка ПМО БиоАПИ может отправить с помощью связи. Главная конечная точка способна отправлять с помощью связи только сообщения запроса ПМО БиоАПИ и сообщения подтверждения ПМО БиоАПИ, а второстепенная конечная точка способна отправлять с помощью связи только сообщения ответа ПМО БиоАПИ и сообщения уведомления ПМО БиоАПИ. Связь ПМО БиоАПИ устанавливается Если конечная точка ПМО БиоАПИ выступает в роли «главной» относительно такой связи и заранее осведомлена об ИИР конечной точки другой конечной точки ПМО БиоАПИ. Принимающая конечная точка ПМО БиоАПИ для участия в связи, устанавливаемой другой конечной точкой ПМО БиоАПИ, невольно играет роль «второстепенной» относительно такой связи, и может изначально не быть осведомлена об ИИР конечной точки другой конечной точки ПМО БиоАПИ (но получает информацию о нем, как только связь будет установлена). Существование более одной связи ПМО БиоАПИ между двумя конечными точками ПМО БиоАПИ при любом распределении ролей невозможно, однако нет ограничения на количество связей, в которых может участвовать конечная точка ПМО БиоАПИ, играющая любую роль. Возможен отказ конечной точки ПМО

БиоАПИ от участия в связи, устанавливаемой определенной конечной точкой ПМО БиоАПИ, а также постоянный отказ конечной точки от действия в роли «второстепенной».

7.4.4 После каждого сообщения запроса ПМО БиоАПИ, отправленного главной конечной точкой с помощью связи ПМО БиоАПИ, следует соответствующее сообщение ответа ПМО БиоАПИ, отправленного второстепенной конечной точкой с помощью такой связи. Некоторые (но не все) сообщения уведомления ПМО БиоАПИ, отправленные второстепенной конечной точкой с помощью связи, сопровождаются последующим подтверждением конечной точки ПМО БиоАПИ, отправленным главной конечной точкой с помощью этой связи. Незапрашиваемые ответы (не имеющие предшествующего запроса) и незапрашиваемые подтверждения (не имеющие предшествующего уведомления) не допускаются. Примерами сообщений запроса ПМО БиоАПИ являются сообщения, поступающие от вызова функции **BioAPI_BSPLoad** или **BioAPI_VerifyMatch,** совершенного локальным приложением в пределах главной конечной точки. Примерами сообщений уведомления ПМО БиоАПИ являются сообщения, возникающие в результате вызова к обработчику событий модуля локальной инфраструктуры или обработчику событий графического интерфейса пользователя, сделанного локальным ПБУ в пределах второстепенной конечной точки.

7.4.5 В настоящем пункте приведен примерт возможного взаимодействия между двумя связанными конечными точками ПМО БиоАПИ, которые содержат поддерживающую инфраструктуру ПМО БиоАПИ. Вызов БиоАПИ, сделанный приложением в главной конечной точке и направленный к ПБУ во второстепенной конечной точке, обычно приводит к сообщению запроса ПМО БиоАПИ, которое отправляется главной инфраструктурой во второстепенную инфраструктуру. Второстепенная инфраструктура обрабатывает сообщение так же, как и при получении вызова собственной функции БиоАПИ ПИП, и после обработки отправляет сообщение ПМО БиоАПИ в главную инфраструктуру,

которая затем возвращает управление локальному приложению. Обратный вызов уведомления (такой как уведомление о событиях модуля и уведомление о событиях графического интерфейса пользователя), совершенный ПБУ во второстепенной конечной точке к инфраструктуре данной точки, обычно приводит к сообщению уведомления ПМО БиоАПИ, которое отправляется второстепенной инфраструктурой в главную инфраструктуру. Главная инфраструктура обрабатывает сообщение, выполнив обратный вызов к функции обработчика, который предоставляется локальным приложением. Для некоторых уведомлений главная инфраструктура отправляет сообщение подтверждения во второстепенную инфраструктуру, которая в таком случае предоставляет контроль локальному ПБУ. Для остальных уведомлений второстепенная инфраструктура предоставляет контроль локальному ПБУ, как только сообщение уведомления ПМО БиоАПИ будет отправлено в главную инфраструктуру.

7.5 Привязка транспортного протокола

7.5.1 Привязка транспортного протокола – это физическая реализация канала связи (либо канала связи запрос/ответ либо канала связи уведомление/подтверждение). В большинстве случаев спецификация привязки относится к обычно применяемым транспортным протоколам вместо определения нового. Спецификация привязки должна обладать исчерпывающими данными о:

- транспортном протоколе;
- выборе параметров и вариантов транспортного протокола;
- кодировании битового уровня сообщений ПМО БиоАПИ;
- адресации механизмов или соглашений;
- возможном взаимном разделении связи транспортного уровня среди множества абстрактных каналов связи;
- опрелениях сообщений транспортного уровня, которые переносят закодированные сообщения ПМО БиоАПИ;

- возможной группировке множества закодированных сообщений ПМО БиоАПИ в одно сообщение транспортного уровня;

- любых дополнительных механизмах поддержки безопасности, надежности, маршрутизации и т. д.

7.5.2 Связь ПМО БиоАПИ содержит канал связи запрос/ответ и может либо не содержать, либо содержать канал связи уведомление/подтверждение. Если у связи ПМО БиоАПИ отсутствует канал связи уведомление/подтверждение, то второстепенная конечная точка не может отправлять сообщение уведомления ПМО БиоАПИ в главную конечную точку. Если связь ПМО БиоАПИ содержит оба канала, эти два канала могут использовать или одну и ту же или разные привязки. В большинстве случаев у связи ПМО БиоАПИ есть оба канала, и эти два канала используют одну и ту же привязку. Однако возможность использования других привязок для этих двух каналов (либо полный отказ от канала уведомление/подтверждение) полезна во многих ситуациях, когда затраты и отдача от затрат связаны с использованием специфической привязки каждым каналом связи между двумя заданными конечными точками ПМО БиоАПИ.

7.5.3 Реализация, соответствующая требованиям настоящего стандарта (при любом классе соответствия), может использовать любую привязку транспортного протокола (либо стандартизированную, либо частную) для каждого канала связи, устанавливающего связь ПМО БиоАПИ с одной конечной точкой ПМО БиоАПИ, при условии, что другая конечная точка ПМО БиоАПИ поддерживает такую привязку транспортного протокола. Некоторые стандартизированные привязки приведены в приложениях А – С с целью облегчения взаимодействия, но не устанавливает обязательного требования к поддержке реализацией любого из них.

7.6 Создание и разрушение связей ПМО БиоАПИ

7.6.1 Связи ПМО БиоАПИ могут быть установлены либо как:

а) управляемые связи или

b) автоматически устанавливаемые связи с помощью так называемых механизмов «Plug and Play» (PnP).

Примечание 1 – Определенная привязка транспортного протокола, описанная в приложениях, определяет стандартный механизм PnP для рассматриваемой привязки транспортного протокола.

Примечание 2 – Настоящий стандарт не распространяется на спецификацию механизмов, используемых для установления связи.

7.6.2 Механизмы PnP, описанные в приложениях, обеспечивают возможность выбора услуг для главных конечных точек ПМО БиоАПИ. Механизмы объявления услуг для второстепенных конечных точек также определены, но их использование является опциональным. Механизмы PnP содержат определения того, как использовать общие механизмы безопасности. Настоящий стандарт не распространяется на спецификацию планируемых к использованию механизмов безопасности.

7.6.3 Сообщения протокола о создании и разрушении связи ПМО БиоАПИ могут передаваться отдельно от сообщений ПМО БиоАПИ, в зависимости от способностей конкретной привязки протокола.

Пример – TCP/IP обеспечивает идентификацию типа сообщений между одними и теми же конечными точками, используя различные номера портов для различных типов сообщений. При привязке TCP/IP к ПМО БиоАПИ используются различные порты для механизма PnP и для сообщений ПМО БиоАПИ.

7.6.4 Поддерживающая инфраструктура ПМО БиоАПИ может явно или неявно создавать связь ПМО БиоАПИ. Явная связь ПМО БиоАПИ создается инфраструктурой в случаях, когда инфраструктура обрабатывает входящий вызов **BioAPI_LinkToEndpoint,** сделанный локальным приложением. При этом конечная точка ПМО БиоАПИ, идентифицированная ИИР конечной точки, который определен приложением, становится второстепенной конечной точкой в новой созданной связи ПМО БиоАПИ, а локальная конечная точка становится главной конечной точкой. Неявная связь ПМО БиоАПИ создается

инфраструктурой не зависимо от того, какие действия выполняет инфраструктура, обрабатывая входящий вызов **BioAPI_Init** или **BioAPI_InitEndpoint**. В этом случае у локального приложения нет прямого контроля над созданием связи ПМО БиоАПИ (инфраструктура выбирает ряд конечных точек ПМО БиоАПИ и пытается установить связь с каждой из них). Существует возможность использовать указанные два метода в комбинации. Обычно, инфраструктура выбирает привязку(и) при использовании каждого канала связи, но приложение может предложить определенную привязку или определенные параметры привязки через вспомогательный параметр функции **BioAPI_LinkToEndpoint** (формат этого вспомогательного параметра не стандартизирован). Связь ПМО БиоАПИ разрушается, когда инфраструктура обрабатывает входящий вызов **BioAPI_UnlinkFromEndpoint** (который определяет ИИР конечной точки), или когда инфраструктура обрабатывает вызов **BioAPI_Terminate** (разрушение всех существующих связей), или когда инфраструктура получает сообщение уведомления ПМО БиоАПИ **masterDeletionEvent** от второстепенной конечной точки или отправляет сообщение уведомления ПМО БиоАПИ **masterDeletionEvent** в главную конечную точку (разрушается связь с такой конечной точкой ПМО БиоАПИ), или когда обнаружен отказ связи транспортного уровня.

7.6.5 Создание связи ПМО БиоАПИ происходит в два этапа. На первом (опциональном) этапе связь транспортного уровня устанавливается для каждого канала (одного или двух) новой связи ПМО БиоАПИ. Перввый этап может быть пропущен, если может быть использована существующая связь транспортного уровня или отсутствует необходимость создания постоянной связи транспортного уровня. На данном этапе назначаются роли главной и второстепенной для соответствующих двух конечных точек (инициатор связи становится главной конечной точкой). На втором этапе главная инфраструктура посылает сообщение запроса ПМО БиоАПИ **addMaster** во второстепенную конечную точку. После получения сообщения запроса ПМО БиоАПИ

addMaster от главной конечной точки второстепенная инфраструктура добавляет ИИР новой главной конечной точки в свою внутреннюю таблицу **VisibleEndpoints** и возвращает на главную конечную точку в сообщении ответа ПМО БиоАПИ **addMaster** информацию, содержащую схему инфраструктуры, список схем ПБУ и список схем ПБФ, скопированных со своего локального реестра компонентов. При получении от второстепенной сообщения ответа ПМО БиоАПИ **addMaster** главная инфраструктура добавляет схему инфраструктуры в свою внутреннюю таблицу **VisibleEndpoints**, а также схемы ПБУ во внутреннюю таблицу **VisibleBSPRegistrations** и схемы ПБФ во внутреннюю таблицу **VisibleBFPRegistrations**. Указанные три таблицы содержат консолидированную информацию схемы, которая поступает из реестров компонентов всех второстепенных конечных точек и от локального реестра компонентов, с ними «консультируются» (вместо локального реестра компонентов) функции **BioAPI_EnumFrameworks, BioAPI_EnumBSPs, BioAPI_EnumBFPs, BioAPI_BSPLoad, BioAPI_BSPAttach** и т. д. Указанные таблицы обновляются не только в случае создания или разрушения связи с второстепенными конечными точками, но также Если происходят какие-либо изменения в реестре компонентов любой второстепенной конечной точки или в локальном реестре компонентов.

7.6.6 Разрушение связи ПМО БиоАПИ также происходит в два этапа. На первом этапе сообщение запроса ПМО БиоАПИ **deleteMaster,** полученное второстепенной конечной точкой от главной, вызывает удаление содержания таблицы **MasterEndpoints,** относящееся к данной главной точке. Если второстепенная по какой-либо причине прекратит работу, она пошлет сообщение уведомления ПМО БиоАПИ **masterDeletionEvent** на все свои главные конечные точки. В каждом случае главная конечная точка удаляет все записи, имеющие отношение к второстепенной конечной точке в своих таблицах **VisibleEndpoints, VisibleBSPRegistrations** и

VisibleBFPRegistrations. На втором (опциональном) этапе связь(и) транспортного уровня, соответствующая(ие) каналу(ам) связи, разрушается любой конечной точкой ПМО БиоАПИ. Второй этап может быть пропущен, если возможность использования этой же связи транспортного уровня для связи ПМО БиоАПИ желательна для каждой конечной точки ПМО БиоАПИ в будущем. Если конечная точка ПМО БиоАПИ обнаруживает отказ связи транспортного уровня связи, из ее таблиц будут удалены все записи, имеющие отношение к составляющей такую связь конечной точке (либо главной, либо второстепенной), а любые поступающие вызовы или обратные вызовы функции, связанные с какой-либо из этих записей, возвратятся с ошибкой.

7.6.7 По способу передачи информационной схемы от второстепенной конечной точки к главной конечной точке, связь ПМО БиоАПИ не является ни симметрической, ни переходной. Например, если конечная точка А связана с конечной точкой В, приложение в конечной точке А обнаруживает ПБУ, зарегистрированные в конечной точке В, но приложение в точке В не будет видеть ПБУ, зарегистрированные в точке А, если другая (независимая) связь не будет установлена от точки В до точки А. Если конечная точка А связана с конечной точкой В, которая связывается с конечной точкой С, приложение в конечной точке А обнаруживает ПБУ, зарегистрированные только в конечной точке В, но не зарегистрированные в коненчой точке С, если другая (независимая) связь не будет установлена от А до С. Утверждение нетранзитивности относится только к реализациям, соответствующих 2-му уровню, но не 1-го уровня, таким как модуль доступа и система контроля доступа, действующих в роли «второстепенной», так как соответствие таких реализаций определяется только по отношению к последовательности сообщений ПМО БиоАПИ, обмен которыми происходит с главной конечной точкой. Модуль доступа или система контроля доступа фактически могут нарушить требование транзитивности, пока они производят сообщения ПМО БиоАПИ, которые могли бы быть произведены ПМО БиоАПИ-

поддерживающей инфраструктурой с подходящим реестром компонентов и набором местных ПБУ и ПБФ.

8 Уведомление о событиях удаленного графического интерфейса пользователя

8.1 ПМО БиоАПИ допускает запуск приложений на компьютере, на который не загружен ПБУ, при этом к компьютеру, на который установлен ПБУ, обычно физически присоединен сканер. Если «ПМО БиоАПИ неосведомленное» приложение БиоАПИ, работающее на установленном ПМО БиоАПИ, запрашивает функцию обратного вызова уведомления о событиях удаленного графического интерфейса пользователя (далее – ГИП), и ПБУ поддерживает выполнение такой функции, то обратный вызов будет направлен в приложение. При получении вышеуказанных обратных вызовов приложение может отразить ГИП на компьютере, где оно запущено, при этом к компьютеру не обязательно должен быть подключен сканер, на котором используется предмет. Любое решение этой проблемы приведет к некоторым изменениям приложения. Тем не менее, ПМО БиоАПИ поддерживает управление уведомлениями ГИП многими способами, включая поддержку двойного (или даже множественного) представления ГИП на различных компьютерах, когда за предметом наблюдает человек, наблюдающий за сбором данных.

8.2 Обратные вызовы уведомлений ГИП отправляются в приложение, которое предназначено для получения таких сообщений путем уточнения типа обратных вызовов ГИП, имеющих отношение к текущей сессии (или к данным ПБУ) и которое впоследствии вызывает функцию сбора данных в пределах той же текущей сессии (или в пределах любой текущей сессии указанного ПБУ).

8.3 В ПМО БиоАПИ указанные обратные вызовы уведомлений ГИП называют первичными. В дополнение к первичным обратным вызовам уведомлений ГИП, ПМО БиоАПИ также поддерживает вторичные обратные вызовы уведомлений ГИП, которые отправляются в подписавшееся ранее приложение (в любую конечную точку) путем предоставления УУИД подписки

на события ГИП. (При наличии подписки на первичные обратные вызовы уведомления ГИП, УУИД подписки на события ГИП отсутствуют).

8.4 Вторичные обратные вызовы уведомлений ГИП содержат те же формы, как и первичные обратные вызовы уведомлений ГИП. Кроме того, одна и та же функция БиоАПИ используется с целью получение первичныч и вторичных обратных вызовов уведомлений ГИП.

8.5 Существует два способа отправления в приложение (называемое приложением обработчика ГИП) обратного вызова уведомлений ГИП:

a) при получении первичного обратного вызова уведомления ГИП приложение осуществляет вызов функции БиоАПИ **BioAPI_NotifyGUIEvent**; параметры этой функции включают в себя все параметры обратного вызова уведомления ГИП, включая УУИД подписки на операции ГИП; такие функции запрашивают инфраструктуру для генерации обратного вызова уведомления ГИП и отправления его в приложение, идентифицированное заданным УУИД подписки на операции ГИП (то есть параметры такого запроса идентифицируют конечную точку, ПБУ, модуль идентификатора устройств и обработчик уведомлений ГИП);

b) перед получением первичного обратного вызова уведомления ГИП приложение вызовает функцию БиоАПИ **BioAPI_RedirectGUIEvents** путем предоставления УУИД подписки на операции ГИП; другие параметры этой функции позволяют приложению указывать вид уведомлений операций ГИП, которые должны быть переадресованы вторичному обработчику ГИП.

9 Примеры возможных конфигураций системы

9.1 Три конечные точки ПМО БиоАПИ представлены на рисунке 1. При стандартном выполнении БиоАПИ любое приложение системы может взаимодействовать с любым ПБУ этой же системы. Данное условие также выполняется при использовании ПМО БиоАПИ.

9.2 Любое приложение, использующее ПМО БиоАПИ, одной системы (конечная точка А) может взаимодействовать с любым ПБУ другой системы

(конечные точки B и C), соблюдающей реализованную в конечных точках удаленную политику доступа.

9.3 В настоящей версии ПМО БиоАПИ отсутствует поддержка взаимодействия приложения конечной точки B с ПБУ конечной точки C (и наоборот), если не будет установлена прямая логическая связь между конечными точками B и C. Таким образом, ретрансляция не поддерживается настоящей версией ПМО БиоАПИ.

9.4. Возможные варианты архитектуры, использующей реализацию 2-го уровня ПМО БиоАПИ представлены на рисунках 2-7.

Примечание – Рисунки 2 - 7 приведены в качестве наглядной иллюстрации и не являются исчерпывающими: существует возможность множества других комбинаций в пределах общей архитектуры, представленной на рисунке 1.

9.5. Конечная точка ПМО БиоАПИ, содержащая приложение с локально загруженным хранилищем ПБУ, и удаленная конечная точка ПМО БиоАПИ, содержащая ПБУ, который обеспечивает работу биометрического сканера представлены на рисунках 2 и 3; на одном сравнение данных реализуется в приложении локально (см. рисунок 2), на другом – удаленно (см. рисунок 3).

9.6. Конечная точка ПМО БиоАПИ, содержащая приложение с локально подключенным биометрическим сканером, и удаленная конечная точка ПМО БиоАПИ, содержащая ПБУ с поддержкой хранилища представлены на рисунках 4 и 5; на одном сравнение данных реализуется в приложении локально (см. рисунок 4), на другом – удаленно (см. рисунок 5).

9.7. Конечная точка ПМО БиоАПИ, содержащая приложение без локально загруженного хранилища ПБУ представлена на рисунках 6 и 7, при этом указанная точка сообщается с удаленными конечными точками, которые обеспечивают сохранение и работу биометрического сканера; на обоих рисунках сравнение данных включается любой из двух удаленнных конечных точек.



Рисунок 1 – Связь приложения с двумя другими ПБУ на двух удаленных системах

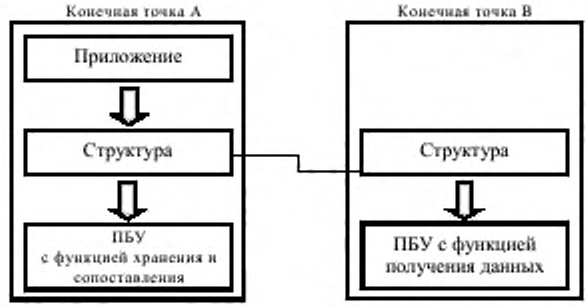


Рисунок 2 – Связь приложения с ПБУ, обеспечивающим получение данных; остальные функции выполняются локально

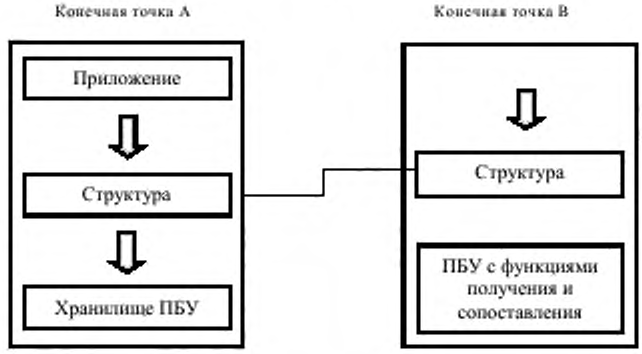


Рисунок 3 – Связь приложения с ПБУ с функциями получения и сопоставления, и локальным хранилищем

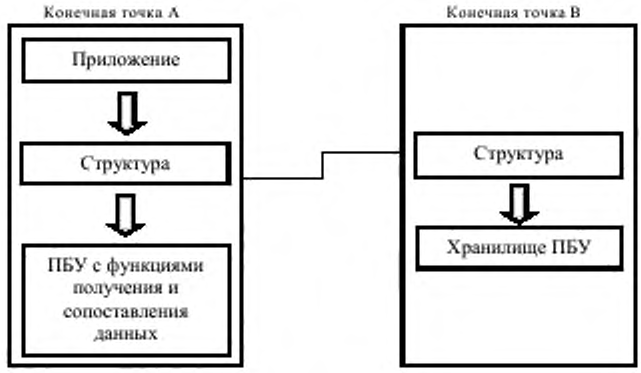


Рисунок 4 – Связь приложения с удаленным хранилищем ПБ; остальными функции приложения выполняются локально

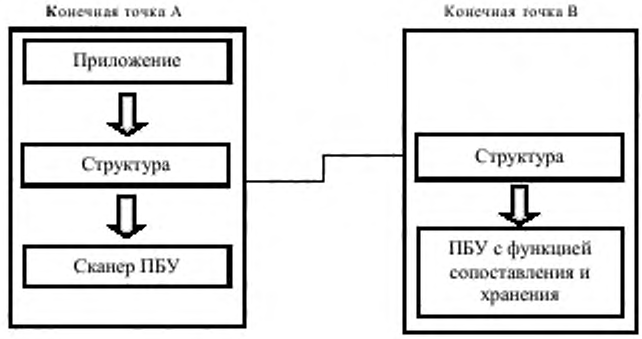


Рисунок 5 – Связь приложения с удаленным ПБУ, обеспечивающем хранение и сопоставление, при локальном сканере



Рисунок 6 – Связь приложения с двумя удаленными ПБУ, одно из которых обеспечивает хранением и сопоставление, а другое обеспечивает получение данных

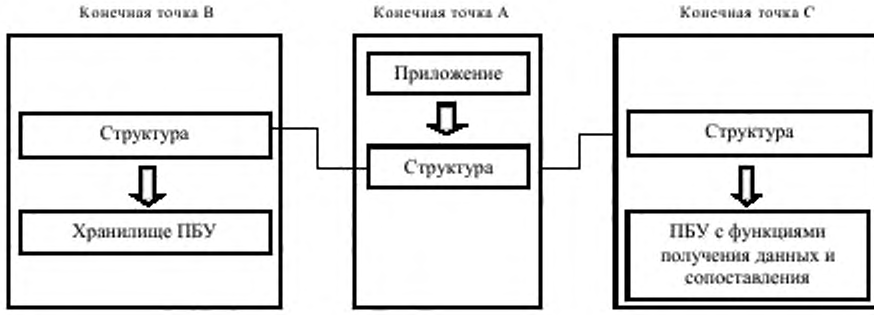


Рисунок 7 – Связь приложения с двумя удаленными ПБУ, одно из которых обеспечивает хранение, а другое – получение данных и их сопоставление

9.8 ПМО БиоАПИ может поддерживать множество других конфигураций хранения, сравнения и получения данных ПБУ. По существу, ПМО БиоАПИ обеспечивает прозрачное (для приложения и ПБУ) расширение инфраструктуры БиоАПИ через границы процесса или узла.

10 Форматы ПМО БиоАПИ

10.1 Многие типы сообщений ПМО БиоАПИ, определенные в настоящем стандарте, могут содержать одну и более ЗБИ. При передаче таких сообщений ПМО БиоАПИ могут быть представлены в любом формате ведущей организации единой структуры форматов обмена биометрическими данными (ЕСФОБД), который должен соответствовать следующим требованиям (см. 13.16):

a) поддерживать как минимум все элементы данных и все абстрактные значения, которые поддерживает формат ведущей организации БиоАПИ;

b) Если он поддерживает дополнительные элементы данных, то все они должны быть опциональными; и

c) в случае сложного формата ведущей организации, он должен быть способен переносить данные, которые включают в себя только простую ЗБИ.

10.2 Любая привязка транспортного протокола определяет, какой формат ведущей организации должен быть применен при использовании такой

привязки. В большинстве случаев это будет или формат ведущей организации БиоАПИ, определенный в ИСО/МЭК 19784-1, или XML формат ведущей организации, определенный в ИСО/МЭК 19785-3.

11 Идентификация конечных точек ПМО БиоАПИ, приложений и ПБУ

11.1 В разделе 10 ИСО/МЭК 19784-1 установлено использование УУИД для идентификации локальной инфраструктуры БиоАПИ и локального ПБУ в качестве продуктов программного обеспечения. Для выявления многократных установок одного и того же продукта программного обеспечения на различных компьютерах, таких УУИД недостаточно, также очевидно, что они не могут выявить ни два случая установки одного и того же ПБУ, который доступен для загрузки в двух различных конечных точках ПМО БиоАПИ на одном и том же компьютере, ни два случая установки инфраструктуры, общей для двух различных конечных точек ПМО БиоАПИ на одном и том же компьютере. По этим причинам, а также в целях обеспечения более гибкой и благоприятной для сети системы обозначения конечных точек ПМО БиоАПИ в настоящем стандарте определено использование ИИР для идентификации конечных точек ПМО БиоАПИ и использование УУИД доступа ПБУ (как альтернативу УУИД продукта ПБУ) для идентификации загружаемых или выполняющихся ПБУ с конечной точкой ПМО БиоАПИ.

11.2 Так как конечная точка ПМО БиоАПИ содержит хотя бы одно приложение, то ИИР конечной точки достаточно, чтобы идентифицировать и конечную точку ПМО БиоАПИ и локальное приложение в такой конечной точке (если оно имеется). ИИР главной конечной точки, который неявно присутствует во всех сообщениях запроса ПМО БиоАПИ, идентифицирует запрашивающее приложение, а ИИР главной конечной точки, который неявно присутствует во всех сообщениях ответа ПМО БиоАПИ, идентифицирует приложение, вызов функции которого возвращается.

11.3 В БиоАПИ каждый ПБУ идентифицирован УУИД. В целях ПМО БиоАПИ такой УУИД упоминается как УУИД продукта ПБУ, чтобы подчеркнуть, что этот УУИД идентифицирует ПБУ как продукт, а не как установленный экземпляр или как исполняемый экземпляр такого продукта (Если один и тот же ПБУ будет установлен на различных компьютерах, на всех компьютерах у ПБУ будет один и тот же УУИД продукта ПБУ). УУИД продукта ПБУ находится в схеме ПБУ и доступен для всех приложений с помощью вызова функции **BioAPI_EnumBSPs.**

11.4 В результате соединения конечных точек ПМО БиоАПИ может получиться, что один и тот же ПБУ будет доступен для загрузки в двух или более различных конечных точках. Хотя УУИД продукта ПБУ уникально идентифицирует ПБУ в пределах конечной точки (так как один и тот же ПБУ не может быть зарегистрирован дважды в одном и том же реестре компонентов), число независимых регистраций одного и того же ПБУ в различных реестрах компонентов (и, таким образом, в различных конечных точках) не ограничено. Это подразумевает, что УУИД продукта ПБУ не всегда предоставляет полную информацию, как функция **BioAPI_BSPLoad.**

11.5 Функция **BioAPI_BSPLoad** рассматривает УУИД как принимаемые данные. В БиоАПИ это должен быть УУИД ПБУ. ПМО БиоАПИ расширяет семантику **BioAPI_BSPLoad,** включая в нее или УУИД продукта ПБУ или другой УУИД, который идентифицирует и ПБУ и конечную точку, в которой он зарегистрирован. Последнее называют УУИД доступа ПБУ и динамически производится главной инфраструктурой для всех (локальных и удаленных) ПБУ.

11.6 В отличие от УУИД продукта ПБУ, который может быть определен приложением из внешнего источника, УУИД доступа ПБУ является динамическим и приложение не может его определить (надежным способом) до времени выполнения. Приложение может также выполнить следующие действия:

a) вызвать **BioAPI_EnumBSPs** и проинспектировать возвращенные схемы ПБУ; для каждого удаленного ПБУ, зарегистрированного и доступного для загрузки во второстепенной конечной точке, схема ПБУ будет содержать УУИД продукта ПБУ, ИИР конечной точки и УУИД доступа ПБУ, а таккже другие данные; для каждого локального ПБУ (зарегистрированного и доступного для загрузки в главной конечной точке), схема ПБУ будет содержать УУИД продукта ПБУ, пустой ИИР конечной точки, и УУИД доступа ПБУ (среди прочих данных); приложение может выбрать поле схемы ПБУ, основанное на УУИД продукта ПБУ и ИИР конечной точки, затем считать УУИД доступа ПБУ из поля этой схемы ПБУ, после чего предоставить УУИД доступа ПБУ в качестве принимаемых данных для **BioAPI_BSPLoad, BioAPI_BSPAttach,** и т. д. или

b) предположить, что требуемый ПБУ точно зарегистрирован в одной из различных доступных конечных точек ПМО БиоАПИ (главная конечная точка и все второстепенные конечные точки), и предоставить (известный) УУИД продукта ПБУ в качестве принимаемых данных для **BioAPI_BSPLoad.**

11.7 Метод, указанный в 11.6, перечисление b), будет работать только в том случае, если предположение будет правильным, но он выгоден тем, что максимизирует возможность успешного запуска неосведомленного приложения ПМО БиоАПИ при установке ПМО БиоАПИ. Даже при наличии множества доступных конечных точек указанный метод может сработать при условии, что никакой ПБУ (с данным УУИД продукта ПБУ) не доступен в более чем одной конечной точке. В противном случае, **BioAPI_BSPLoad** возвратит ошибку с указанием, что предоставленный УУИД продукта ПБУ неоднозначен и вызов должен быть повторен с целью получения УУИД доступа ПБУ взамен.

12 Краткий обзор обменов ПМО БиоАПИ

12.1 Обеспечение безопасности и конфиденциальности

ПМО БиоАПИ не обеспечивает условий безопасного и конфиденциального обмена. Безопасности и конфиденциальность обеспечиваются путем использования привязок транспортного протокола, которые предоставляют механизмы безопасности.

12.2 Вызов приложением функций на удаленном ПБУ

В Таблице 3 приведены функции БиоАПИ ПИП, с помощью которых выполняются сообщения запроса ПМО БиоАПИ, ссылки на ИСО/МЭК 19784-1, которые определяют функции БиоАПИ, соответствующие сообщения запроса ПМО БиоАПИ и пункты настоящего стандарта, которые определяют сообщения ПМО БиоАПИ.

Таблица 3 - Корреспонденция между функциями БиоАПИ и типами сообщения ПМО БиоАПИ

Функция BioAPI	Пункт БиоАПИ	Типы сообщения ПМО БиоАПИ	Ссылка
BioAPI_Init	8.1.1	Отсутствует	16.1
BioAPI_InitEndpoint		Отсутствует	16.2
BioAPI_Terminate	8.1.2	Уведомление masterDeletionEvent	16.3
BioAPI_LinkToEndpoint		Запрос и ответ addMaster	16.4
BioAPI_UnlinkFromEndpoint		Запрос и ответ deleteMaster	16.5
BioAPI_EnumFrameworks		Отсутствует	16.6
BioAPI_EnumBSPs	8.1.4	Отсутствует	16.7
BioAPI_EnumBFPs	8.1.10	Отсутствует	16.8
BioAPI_BSPLoad	8.1.5	Запрос и ответ bspLoad	16.9
BioAPI_BSPUnload	8.1.6	Запрос и ответ bspUnload	16.10
BioAPI_QueryUnits	8.1.9	Запрос и ответ queryUnits	16.11
BioAPI_QueryBFPs	8.1.11	Запрос и ответ queryBFPs	16.12
BioAPI_BSPAttach	8.1.7	Запрос и ответ bspAttach	16.13
BioAPI_BSPDetach	8.1.8	Запрос и ответ bspDetach	16.14
BioAPI_EnableEvents	8.3.1	Запрос и ответ enableUnitEvents	16.15
BioAPI_EnableEventNotifications		Запрос и ответ enableEventNotifications	16.16
BioAPI_ControlUnit	8.1.12	Запрос и ответ controlUnit	16.17
BioAPI_Control		Запрос и ответ управления	16.18
BioAPI_FreeBIRHandle	8.2.1	Запрос и ответ freeBIRHandle	16.19
BioAPI_GetBIRFromHandle	8.2.2	Запрос и ответ getBIRFromHandle	16.20
BioAPI_GetHeaderFromHandle	8.2.3	Запрос и ответ getHeaderFromHandle	16.21
BioAPI_SubscribeToGUIEvents		Запрос и ответ subscribeToGUIEvents	16.22
BioAPI_UnsubscribeFromGUIEvents		Запрос и ответ unsubscribeFromGUIEvents	16.23
BioAPI_QueryGUIEventSubscriptions		Запрос и ответ queryGUIEventSubscriptions	16.24
BioAPI_NotifyGUISelectEvent		Запрос и ответ notifyGUISelectEvent	16.25
BioAPI_NotifyGUIStateEvent		Запрос и ответ notifyGUIStateEvent	16.26
BioAPI_NotifyGUIProgressEvent		Запрос и ответ notifyGUIProgressEvent	16.27
BioAPI_RedirectGUIEvents		Запрос и ответ redirectGUIEvents	16.28

Продолжение таблицы 3

Функция BioAPI	Пункт БиоАПИ	Типы сообщения ПМО БиоАПИ	Ссылка
BioAPI_UnredirectGUIEvents		Запрос и ответ unredirectGUIEvents	16.29
BioAPI_Capture	8.4.1	Запрос на сбор данных и ответ	16.30
BioAPI_CreateTemplate	8.4.2	Запрос createTemplate и ответ	16.31
BioAPI_Process	8.4.3	Запрос и ответ регистрации	16.32
BioAPI_ProcessWithAuxBIR	8.4.4	Запрос и ответ processWithAuxBIR	16.33
BioAPI_VerifyMatch	8.4.5	Запрос и ответ VerifyMatch	16.34
BioAPI_IdentifyMatch	8.4.6	Запрос и ответ identifyMatch	16.35
BioAPI_Enroll	8.4.7	Запрос и ответ регистрации	16.36
BioAPI_Verify	8.4.8	Запрос и ответ проверки	16.37
BioAPI_Identify	8.4.9	Запрос и ответ идентификации	16.38
BioAPI_Import	8.4.10	Запрос и ответ импорта	16.39
BioAPI_PresetIdentifyPopulation	8.4.11	Запрос и ответ presetIdentifyPopulation	16.40
BioAPI_Transform		Запрос и ответ преобразования	16.41
BioAPI_DbOpen	8.5.1	Запрос и ответ dbOpen	16.42
BioAPI_DbClose	8.5.2	Запрос и ответ dbClose	16.43
BioAPI_DbCreate	8.5.3	Запрос и ответ dbCreate	16.44
BioAPI_DbDelete	8.5.4	Запрос и ответ dbDelete	16.45
BioAPI_DbSetMarker	8.5.5	Запрос и ответ dbSetMarker	16.46
BioAPI_DbFreeMarker	8.5.6	Запрос и ответ dbFreeMarker	16.47
BioAPI_DbStoreBIR	8.5.7	Запрос и ответ dbStoreBIR	16.48
BioAPI_DbGetBIR	8.5.8	Запрос и ответ dbGetBIR	16.49
BioAPI_DbGetNextBIR	8.5.9	Запрос и ответ dbGetNextBIR	16.50
BioAPI_DbDeleteBIR	8.5.10	Запрос и ответ dbDeleteBIR	16.51
BioAPI_CalibrateSensor	8.6.4	Запрос и ответ calibrateSensor	16.52
BioAPI_SetPowerMode	8.7.1	Запрос и ответ setPowerMode	16.53
BioAPI_SetIndicatorStatus	8.6.2	Запрос и ответ setIndicatorStatus	16.54
BioAPI_GetIndicatorStatus	8.6.3	Запрос и ответ getIndicatorStatus	16.55
BioAPI_Cancel	8.7.1	Запрос и ответ отмены	16.57
BioAPI_Free	8.7.2	Отсутствует	16.58
BioAPI_RegisterBSP		Запрос registerBSP, ответ registerBSP и уведомление bspRegistrationEvent	16.59
BioAPI_UnregisterBSP		Запрос unregisterBSP, ответ unregisterBSP, и уведомление bspUnregistrationEvent	16.60

Окончание таблицы 3

Функция BioAPI	Пункт БиоАПИ	Типы сообщения ПМО БиоАПИ	Ссылка
BioAPI_RegisterBFP		Запрос registerBFP, ответ registerBFP, и уведомление bfpRegistrationEvent	16.61
BioAPI_UnregisterBFP		Запрос unregisterBFP, ответ unregisterBFP, и уведомление bfpUnregistrationEvent	16.62
BioAPI_EVENT_HANDLER	7.28	Уведомление unitEvent	17.1
BioAPI_GUI_SELECT_EVENT_HANDLER		Уведомление и подтверждение guiSelectEvent	17.2
BioAPI_GUI_STATE_EVENT_HANDLER		Уведомление и подтверждение guiStateEvent	17.3
BioAPI_GUI_PROGRESS_EVENT_HANDLER		Уведомление и подтверждение guiProgressEvent	17.4

12.3 Обращение приложения к функциям, не имеющим связанных с ними сообщений ПМО БиоАПИ

Небольшое количество запускаемых приложениями функций БиоАПИ ПИП не идентифицируют ПБУ и они являются локальными. Данные функции не выполняют корреспондирующие обмены ПМО БиоАПИ и разрешаются путем информации, доступной для локальной инфраструктуры. Эти функции БиоАПИ указаны в таблице 4.

Таблица 4 – Функции БиоАПИ без соответствующего типа сообщения ПМО БиоАПИ

Функция BioAPI	Пункт BioAPI	Источник информации	Ссылка
BioAPI_EnumFrameworks		Таблица VisibleEndpoints	18.2
BioAPI_EnumBSPs	8.1.4	Таблица VisibleBSPRegistrations	18.3
BioAPI_EnumBFPs	8.1.10	Таблица VisibleBFPRegistrations	18.4

Окончание таблицы 4

Функция BioAPI	Пункт BioAPI	Источник информации	Ссылка
BioAPI_Free	8.7.2	Таблица ApplicationOwnedMemoryBlocks	18.13

12.4 Операционные уведомления

Если приложением включены уведомления о событиях модуля (см. 9.2.1 и 10.2.1.1/2 БиоАПИ) конкретного удаленного ПБУ, использующего **BioAPI_EnableEventNotifications**, то события, которые происходят в ПБУ, передаются с помощью сообщений уведомления ПМО БиоАПИ unitEvent в удаленную систему, которая содержит такое приложение (см. 17.1).

13 Общие положения

13.1 Настоящий раздел содержит положения, которые используются в качестве ссылок в других разделах настоящего стандарта.

13.2 Для создания и отправления сообщений запроса ПМО БиоАПИ заданного типа к второстепенной конечной точке с заданным ИИР конечной точки, инфраструктура должна создать временное абстрактное значение типа **BIPMessage** (см. раздел 14), для которого:

a) должен быть выбран альтернативный компонент **request**;

b) компонент **masterEndpointIRI** должен быть установлен на ИИР конечной точки;

c) компонент **slaveEndpointIRI** должен быть установлен на заданный ИИР второстепенной конечной точки;

d) компонент **linkNumber** должен быть установлен на номер связи (см. 16.4.4, перечисление a)), относящийся к связи ПМО БиоАПИ от локальной конечной точки до указанной второстепенной конечной точки;

e) компонент **requestId** должен быть установлен на текущий идентификатор запроса, относящийся к связи ПМО БиоАПИ (см. 16.4.4,

перечисление a)), который затем должен быть (если его значение менее 4294967295) увеличен на единицу или (в противном случае) установлен на ноль;

f) должна быть выбрана альтернатива компонента **params,** корреспондирующая заданному типу сообщения запроса ПМО БиоАПИ и

g) выбранная альтернатива компонента **params** должна быть установлена на значение параметра, определенного в обращении к данному подпункту

и затем отправить (см. 13.8) абстрактное значение **BIPMessage** к заданной второстепенной конечной точке ПМО БиоАПИ.

13.3 Для создания и отправления корреспондирующего сообщения ответа ПМО БиоАПИ на сообщение запроса ПМО БиоАПИ, инфраструктура должна создать временное абстрактное значение типа **BIPMessage** (см. раздел 14), для которого:

a) должен быть выбран альтернативный **response**;

b) компоненты **masterEndpointIRI**, **slaveEndpointIRI**, **linkNumber** и **requestId** должны быть установлены из одноименных компонентов сообщения запроса ПМО БиоАПИ;

c) должна быть выбрана альтернатива для компонента **params** с названием, аналогичным названию альтернативы компонента **params,** который присутствует в сообщении запроса ПМО БиоАПИ;

d) отобранная альтернатива компонента **params** должна быть установлена на значение параметра, определенного в обращении к этому подпункту и

e) компонент **returnValue** должен быть установлен на возвращаемое значение ответа, определенного в обращении к этому подпункту

и затем отправить (см. 13.8) абстрактное значение **BIPMessage** к конечной точке ПМО БиоАПИ, которая идентифицирована значением компонента **masterEndpointIRI.**

13.4 Для создания и отправления сообщения уведомления ПМО БиоАПИ заданного типа к главной конечной точке с заданным ИИР конечной точки, инфраструктура должна создать временное абстрактное значение типа **BIPMessage** (см. раздел 14), для которого:

a) должен быть выбран альтернативный компонент **notification**;

b) компонент **slaveEndpointIRI** должен быть установлен на ИИР локальной конечной точки;

c) компонент **masterEndpointIRI** должен быть установлен на заданный ИИР главной конечной точки;

d) компонент **linkNumber** должен быть установлен на номер связи (см. 15.4.4, перечисление a)), относящийся к связи ПМО БиоАПИ от заданной главной конечной точки до локальной конечной точки;

e) компонент **notifcationId** должен быть установлен на текущей идентификатор уведомления, относящийся к связи ПМО БиоАПИ (см. 16.4.4, перечисление a)), который затем (если его значение менее 4294967295) должен быть увеличен на единицу или (в противном случае) установлен на ноль;

f) должна быть отобрана альтернатива компонента **params**, корреспондирующая соответствующему типу сообщения уведомления ПМО БиоАПИ; и

g) отобранная альтернатива компонента **params** должна быть установлена на значение параметра, определенного в требованиях данного подпункта;

и отправить (см. 13.8) абстрактное значение **BIPMessage** к конечной точке ПМО БиоАПИ, которая идентифицирована значением компонента **masterEndpointIRI.**

13.5 Для создания и отправления сообщения подтверждения ПМО БиоАПИ, корреспондирующего соответствующему сообщению уведомления ПМО БиоАПИ, инфраструктура должна создать временное абстрактное значение типа **BIPMessage** (см. раздел 14), для которого:

a) должен быть отобран альтернативный компонент **acknowledgement;**

b) должны быть установлены компоненты **masterEndpointIRI**, **slaveEndpointIRI**, **linkNumber** и **notifcationId** из одноименных компонентов сообщения уведомления ПМО БиоАПИ;

c) должна быть отобрана альтернатива компонента **params** с названием, аналогичным альтернативе для компонента **params,** который присутствует в сообщении уведомления ПМО БиоАПИ;

d) отобранная альтернатива для компонента **params** должна быть установлена на значение параметра, определенного в обращении к этому подпункту и

e) компонент **returnValue** должен быть установлен на возвращаемое значение, определенное в обращении к этому подпункту;

и затем отправить (см. 13.8) абстрактное значение **BIPMessage** к конечной точке ПМО БиоАПИ, которая идентифицирована значением компонента **slaveEndpointIRI.**

13.6 Прием сообщения ответа ПМО БиоАПИ, корреспондирующего заданному сообщению запроса ПМО БиоАПИ, означает, что инфраструктура получает (см. 13.9) временное абстрактное значение типа **BIPMessage** (см. раздел 14), для которого:

a) присутствует альтернативный компонент **response**;

b) компоненты **masterEndpointIRI**, **slaveEndpointIRI**, **linkNumber** и **requestId** обладают такими же значениями, как и одноименные компоненты сообщения запроса ПМО БиоАПИ; и

c) альтернатива присутствующего компонента **params** имеет название, аналогичное альтернативе для компонента **params,** который присутствует в сообщении ПМО БиоАПИ запроса.

13.7 Прием сообщения подтверждения ПМО БиоАПИ, корреспондирующего заданному сообщению уведомления ПМО БиоАПИ,

означает, что инфраструктура получает (см. 13.9) временное абстрактное значение типа **BIPMessage** (см. раздел 14), для которого:

a) присутствует альтернативный компонент **acknowledgement**;

b) компоненты **masterEndpointIRI**, **slaveEndpointIRI**, **linkNumber** и **notificationId** обладают такими же значениями, как и одноименные компоненты сообщения уведомления ПМО БиоАПИ и

c) альтернатива присутствующего компонента **params** обладает тем же названием, как и альтернатива для компонента **params,** который присутствует в сообщении уведомления ПМО БиоАПИ.

13.8 Термин «отправлять» («send»), использующийся в настоящем стандарте, означает концептуальную передачу сообщения от конечной точки ПМО БиоАПИ (отправитель) в связь ПМО БиоАПИ между конечной точкой отправителя и другой конечной точкой ПМО БиоАПИ (приемник). Передаваемое сообщение является абстрактным значением типа **BIPMessage** (см. раздел 14), которое помещается в связь ПМО БиоАПИ в процессе отправления. Не делается никаких предположений относительно природы связи ПМО БиоАПИ (за исключением того, что она логически соединяет две конечные точки и имеет соответствующую ориентацию), транспортного механизма или используемого протокола кодирования передаваемого сообщения ПМО БиоАПИ и временных или связанных с «местом» аспектам отправления (такими как перерыв и организация очереди).

13.9 Термин «получать» («receive»), использующийся в настоящем стандарте, означает концептуальную передачу сообщения из связи ПМО БиоАПИ между конечной точкой ПМО БиоАПИ (отправитель) и другой конечной точкой ПМО БиоАПИ (приемник) к конечной точке приемника. Передаваемое сообщение является абстрактным значением типа **BIPMessage** (см. раздел 13), которое извлечено из связи ПМО БиоАПИ и скопировано в недавно созданное временное абстрактное значение (см. 13.11) в пределах конечной точки приемника в процессе получения. Не делается никаких

предположений относительно природы связи ПМО БиоАПИ (за исключением того, что она логически соединяет две конечные точки и имеет соответствующую ориентацию), кодирования передаваемого сообщения ПМО БиоАПИ, и временным или связанным с «местом» аспектам получения (такими как ожидание и организация очереди).

Примечание – Данное положение означает, что когда в тексте встречается предложение нормативного характера типа «получение корреспондирующего сообщения ответа ПМО БиоАПИ» (ответ, содержащий идентификатор запроса), реализация инфраструктуры в течение неопределенного промежутка времени возможно будет находится в режиме ожидания до фактического приема сообщения ПМО БиоАПИ. Спецификация привязок, указанная в приложениях А и С представляет собой детальный разбор получения сообщения ПМО БиоАПИ с использованием некоторых видов транспортировки.

13.10 Внутренний запрос функции БиоАПИ – это концептуальный вызов функции, совершенный инфраструктурой к функции собственного интерфейса БиоАПИ таким способом, при котором запрос обрабатывается в соответствии с ИСО/МЭК 19784-1, а не согласно условиям, приведенным в настоящем стандарте, принятым для вызовов той же самой функции БиоАПИ, совершенным локальным приложением. Внутренний запрос функции не обязательно должен быть запросом функции Си (так как отсутствует какое-либо требование для второго особого интерфейса БиоАПИ в инфраструктуре), однако он будет поддерживаться любым механизмом, соответствующим следующим требованиям:

a) прежде чем выполнить внутренний запрос, инфраструктура должна иметь возможность установить параметры внутреннего запроса;

b) выполняя внутренний запрос, инфраструктура должна выполнить все действия, указанные в ИСО/МЭК 19784-1 для входящего вызова функции БиоАПИ, включая определение возвращаемого значения;

c) выполненные действия не должны включать в себя ни одну из модификаций и дополнений, определенных в настоящем стандарте, для

входящего вызова той же самой функции БиоАПИ (таких как определение главенствующей конечной точки) и

d) после того как выполнение внутреннего запроса будет закончено, инфраструктура должна иметь возможность прочитать значение параметра и возвращаемое значение внутреннего запроса.

13.11 В соответствии с требованиями настоящего стандарта должны быть созданы некоторые временные абстрактные значения типа ACH.1, которые рассмотрены далее с определением их использования. Настоящий стандарт не распространяется на удаление временного абстрактного значения. Структура может удалить временное абстрактное значение в любое время со следующими ограничениями:

a) временное абстрактное значение, создаваемое при обработке поступающего запроса функции БиоАПИ от локального приложения, не должно быть удалено до возвращения инфраструктурой управления локальному приложению;

b) временное абстрактное значение, создаваемое при обработке поступающего сообщения запроса ПМО БиоАПИ от главной конечной точки, не должно быть удалено до отправления инфраструктурой корреспондирующего сообщения ответа ПМО БиоАПИ в главную конечную точку;

c) временное абстрактное значение, создаваемое при обработке поступающего обратного вызова от ПБУ, не должно быть удалено до возвращение инфраструктурой управления ПБУ и

d) временное абстрактное значение, создаваемое при обработке поступающего сообщения уведомления ПМО БиоАПИ от второстепенной конечной точки не должно быть удалено до того, как инфраструктура либо отправит корреспондирующее сообщение подтверждения ПМО БиоАПИ во второстепенную конечную точку, либо примет решение не отправлять сообщение подтверждения ПМО БиоАПИ.

13.12 В настоящем стандарте установлено, что при преобразовании АСН.1 в Си должна быть образована переменная некоторого типа Си (или массива октетов, или массива элементов некоторого типа Си), а ее адрес должен быть назначен на другую переменную Си. Настоящий стандарт не распространяется на удаление образованной переменной или массива. Структура может удалить образованную переменную или массив в любое время со следующими ограничениями:

a) образованная переменная или массив, создаваемые при обработке поступающего запроса функции БиоАПИ от локального приложения, никогда не должны удаляться за исключением случаев, когда последующий входящий вызов **BioAPI_Free** от локального приложения.

Примечание 1 – Такие образованные переменные и массивы могут быть только выходными параметрами функций БиоАПИ, которые создаются главной инфраструктурой и возвращаются к локальному приложению;

b) образованная переменная или массив, создаваемые при обработке поступающего сообщения запроса ПМО БиоАПИ от главной конечной точки, не должны быть удалены (см. 13.14) до отправления инфраструктурой корреспондирующего сообщения ответа ПМО БиоАПИ в главную конечную точку.

Примечание 2 – Такие образованные переменные или массивы могут быть только выходными параметрами функций БиоАПИ, которые создаются второстепенной инфраструктурой и возвращаются внутренним запросом;

c) образованная переменная или массив, создаваемые при обработке поступающего обратного вызова от ПБУ, не должны быть удалены до возвращения инфраструктурой контроля ПБУ.

Примечание 3 – Такие образованные переменные или массивы могут быть только входными параметрами функции обратного вызова, которые создаются главной инфраструктурой и предоставляются локальному приложению;

d) образованная переменная или массив, создаваемые при обработке поступающего сообщения уведомления ПМО БиоАПИ от второстепенной конечной точки, не должны быть удалены прежде, чем инфраструктура либо

отправит корреспондирующее сообщение подтверждения ПМО БиоАПИ во второстепенную конечную точку, либо примет решение не отправлять сообщение подтверждения ПМО БиоАПИ.

Примечание 4 – Такие образованные переменные или массивы могут быть только входными параметрами функций обратного вызова, которые создаются основной инфраструктурой и предоставляются локальному приложению.

13.13 При образовании переменной или массива в ходе обработки поступающего запроса функции БиоАПИ от локального приложения (см. 13.12) инфраструктура должна добавить данные, содержащие адрес образованной переменной или массива, в таблицу **ApplicationOwnedMemoryBlocks** (см. 18.13).

13.14 Удаление образованной переменной или массива, созданных при обработке поступающего сообщения запроса ПМО БиоАПИ (см. 13.12, перечисление (b)), должно быть выполнено путем выполнения внутреннего запроса функцией БиоАПИ (см. 13.10) к функции **BioAPI_Free**, в котором параметр запроса функции должен быть адресом образованной переменной или массива.

13.15 Номер версии настоящего стандарта: главное число 1, второстепенное число 0.

13.16 Когда содержащее ЗБИ сообщение ПМО БиоАПИ переносится в пределах сообщения транспортного уровня, ЗБИ может быть в любом формате ведущей организации ЕСФОБД, который соответствует следующим требованиям:

a) поддерживает все элементы данных и все абстрактные значения, которые поддерживает формат ведущей организации ЕСФОБД;

b) если формат поддерживает дополнительные элементы данных, то они являются необязательными и

c) при сложном формате, он должен быть способен переносить данные, которые включают в себя только простую ЗБИ;

и не должен быть ни в одном из форматов ведущей организации, который не отвечают указанным требованиям.

13.17 Если при кодировании на уровне привязки транспортного протокола сообщения ПМО БиоАПИ оно содержит абстрактное значение **BioAPI-BIR-HEADER** или **BioAPI-BIR**, то компонент **formattedBIR** такого абстрактного значения (которое является ЗБИ в формате ведущей организации, определенном в ИСО/МЭК 19784-1, приложение В) может быть преобразован (как случай реализации транспортного протокола) в другой формат ведущей организации, предшествующий кодированию. Такое преобразование допускается только в том случае, если новый формат соответствует требованиям 13.16.

13.18 Если после декодирования обнаруживается (на уровне привязки транспортного протокола), что поступающее сообщение ПМО БиоАПИ содержит абстрактное значение типа **BioAPI-BIR-HEADER** или **BioAPI-BIR,** в котором компоненты **patronFormatOwner** и **patronFormatType** имеют значения, отличающиеся от 257 и 8 (в указанном порядке), то компонент **formattedBIR** такого абстрактного значения должен быть либо преобразован в формат ведущей организации, определенный в ИСО/МЭК 19784-1, приложение В (только если оригинальный формат ведущей организации соответствует требованиям 13.16, и реализации известен способ выполнения преобразования), либо оставлен неизменным. Если преобразование будет выполнено, то компонентам **patronFormatOwner** и **patronFormatType** должны быть присвоены значения 257 и 8 (в указанном порядке).

14 Синтаксис сообщений ПМО БиоАПИ

14.1 Сообщение ПМО БиоАПИ должно состоять из следующих абстрактных значений типа АСН.1 **BIPMessage**:

```
BIPMessage ::= SEQUENCE {
        nature          CHOICE {
        request         BIPRequest,
        response        BIPResponse,
```

```
        notification                BIPNotification,
        acknowledgement             BIPAcknowledgement
        },
    }
BIPRequest ::= SEQUENCE {
        slaveEndpointIRI            EndpointIRI,
        masterEndpointIRI           EndpointIRI,
        linkNumber                  UnsignedInt,
        requestId                   UnsignedInt,
        params                      CHOICE {
            addMaster                   AddMaster-RequestParams,
            deleteMaster                DeleteMaster-RequestParams,
            bspLoad                     BSPLoad-RequestParams,
            bspUnload                   BSPUnload-RequestParams,
            queryUnits                  QueryUnits-RequestParams,
            queryBFPs                   QueryBFPs-RequestParams,
            bspAttach                   BSPAttach-RequestParams,
            bspDetach                   BSPDetach-RequestParams,
            enableUnitEvents            EnableUnitEvents-
                                        RequestParams,
            enableEventNotifications    EnableEventNotifications-
                                        RequestParams,
            controlUnit                 ControlUnit-RequestParams,
            control                     Control-RequestParams,
            freeBIRHandle               FreeBIRHandle-RequestParams,
            getBIRFromHandle            GetBIRFromHandle-
                                        RequestParams,
            getHeaderFromHandle         GetHeaderFromHandle-
                                        RequestParams,
            subscribeToGUIEvents        SubscribeToGUIEvents-
                                        RequestParams,
            unsubscribeFromGUIEvents    UnsubscribeFrom-GUIEvents-
                                        RequestParams,
            redirectGUIEvents           RedirectGUIEvents-
                                        RequestParams,
            unredirectGUIEvents         UnredirectGUIEvents-
                                        RequestParams,
            queryGUIEventSubscriptions  QueryGUIEvent-Subscriptions-
                                        RequestParams,
            notifyGUISelectEvent        NotifyGUISelectEvent-
                                        RequestParams,
```

```
    notifyGUIStateEvent             NotifyGUIStateEvent-
                                    RequestParams,
    notifyGUIProgressEvent          NotifyGUIProgressEvent-
                                    RequestParams,
    capture                         Capture-RequestParams,
    createTemplate                  CreateTemplate-RequestParams,
    process                         Process-RequestParams,
    processWithAuxBIR               ProcessWithAuxBIR-
                                    RequestParams,
    verifyMatch                     VerifyMatch-RequestParams,
    identifyMatch                   IdentifyMatch-RequestParams,
    enroll                          Enroll-RequestParams,
    Verify                          Verify-RequestParams,
    identify                        Identify-RequestParams,
    import                          Import-RequestParams,
    presetIdentifyPopulation        PresetIdentifyPopulation-
                                    RequestParams,
    transform                       Transform-RequestParams,
    dbOpen                          DbOpen-RequestParams,
    dbClose                         DbClose-RequestParams,
    dbCreate                        DbCreate-RequestParams,
    dbDelete                        DbDelete-RequestParams,
    dbSetMarker                     DbSetMarker-RequestParams,
    dbFreeMarker                    DbFreeMarker-RequestParams,
    dbStore                         DbStoreBIR-RequestParams,
    dbGetBIR                        DbGetBIR-RequestParams,
    dbGetNextBIR                    DbGetNextBIR-RequestParams,
    dbDeleteBIR                     DbDeleteBIR-RequestParams,
    calibrateSensor                 CalibrateSensor-RequestParams,
    setPowerMode                    SetPowerMode-RequestParams,
    setIndicatorStatus              SetIndicatorStatus-
                                    RequestParams,
    getIndicatorStatus              GetIndicatorStatus-
                                    RequestParams,
    cancel                          Cancel-RequestParams,
    registerBSP                     RegisterBSP-RequestParams,
    unregisterBSP                   UnregisterBSP-RequestParams,
    registerBFP                     RegisterBFP-RequestParams,
    unregisterBFP                   UnregisterBFP-RequestParams,
    ...
}
```

```
}

BIPResponse ::= SEQUENCE {
        slaveEndpointIRI                EndpointIRI,
        masterEndpointIRI               EndpointIRI,
        linkNumber                      UnsignedInt,
        requestId                       UnsignedInt,
        params                          CHOICE {
            addMaster                   AddMaster-ResponseParams,
            deleteMaster                DeleteMaster-ResponseParams,
            bspLoad                     BSPLoad-ResponseParams,
            bspUnload                   BSPUnload-ResponseParams,
            queryUnits                  QueryUnits-ResponseParams,
            queryBFPs                   QueryBFPs-ResponseParams,
            bspAttach                   BSPAttach-ResponseParams,
            bspDetach                   BSPDetach-ResponseParams,
            enableUnitEvents            EnableUnitEvents-ResponseParams,
            enableEventNotifications    EnableEventNotifica-tions-
                                        ResponseParams,
            controlUnit                 ControlUnit-ResponseParams,
            control                     Control-ResponseParams,
            freeBIRHandle               FreeBIRHandle-ResponseParams,
            getBIRFromHandle            GetBIRFromHandle-ResponseParams,
            getHeaderFromHandle         GetHeaderFromHandle-
                                        ResponseParams,
            subscribeToGUIEvents        SubscribeToGUIEvents-ResponseParams,
            unsubscribeFromGUIEvents    UnsubscribeFromGUIEvents-
                                        ResponseParams,
            redirectGUIEvents           RedirectGUIEvents-ResponseParams,
            unredirectGUIEvents         UnredirectGUIEvents-ResponseParams,
            queryGUIEventSubscriptions  QueryGUIEventSubscrip-tions-
                                        ResponseParams,
            notifyGUISelectEvent        NotifyGUISelectEvent-ResponseParams,
            notifyGUIStateEvent         NotifyGUIStateEvent-ResponseParams,
            notifyGUIProgressEvent      NotifyGUIProgressEvent-
                                        ResponseParams,
            capture                     Capture-ResponseParams,
            createTemplate              CreateTemplate-ResponseParams,
            process                     Process-ResponseParams,
            processWithAuxBIR           ProcessWithAuxBIR-ResponseParams,
            verifyMatch                 VerifyMatch-ResponseParams,
```

```
                identifyMatch               IdentifyMatch-ResponseParams,
                enroll                      Enroll-ResponseParams,
                verify                      Verify-ResponseParams,
                identify                    Identify-ResponseParams,
                import                      Import-ResponseParams,
                presetIdentifyPopulation    PresetIdentifyPopulation-
                                            ResponseParams,
                transform                   Transform-ResponseParams,
                dbOpen                      DbOpen-ResponseParams,
                dbClose                     DbClose-ResponseParams,
                dbCreate                    DbCreate-ResponseParams,
                dbDelete                    DbDelete-ResponseParams,
                dbSetMarker                 DbSetMarker-ResponseParams,
                dbFreeMarker                DbFreeMarker-ResponseParams,
                dbStore                     DbStoreBIR-ResponseParams,
                dbGetBIR                    DbGetBIR-ResponseParams,
                dbGetNextBIR                DbGetNextBIR-ResponseParams,
                dbDeleteBIR                 DbDeleteBIR-ResponseParams,
                calibrateSensor             CalibrateSensor-ResponseParams,
                setPowerMode                SetPowerMode-ResponseParams,
                setIndicatorStatus          SetIndicatorStatus-ResponseParams,
                getIndicatorStatus          GetIndicatorStatus-ResponseParams,
                cancel                      Cancel-ResponseParams,
                registedBSP                 RegisterBSP-ResponseParams,
                unregisterBSP               UnregisterBSP-ResponseParams,
                registerBFP                 RegisterBFP-ResponseParams,
                unregisterBFP               UnregisterBFP-ResponseParams,
                ...
        },
        returnValue                         BioAPI-RETURN
}

BIPNotification ::= SEQUENCE {
        masterEndpointIRI                   EndpointIRI,
        slaveEndpointIRI                    EndpointIRI,
        linkNumber                          UnsignedInt,
        notificationId                      UnsignedInt,
        params                              CHOICE {
                masterDeletionEvent         MasterDeletionEvent-NotificationParams,
                unitEvent                   UnitEvent-NotificationParams,
                guiSelectEvent              GUISelectEvent-NotificationParams,
```

```
                guiStateEvent               GUIStateEvent-NotificationParams,
                guiProgressEvent            GUIProgressEvent-NotificationParams,
                bspRegistrationEvent        BSPRegistrationEvent-
                NotificationParams,
                bspUnregistrationEvent      BSPUnregistrationEvent-
                NotificationParams,
                bfpRegistrationEvent BFPRegistrationEvent-NotificationParams,
                bfpUnregistrationEvent      BFPUnregistrationEvent-
                NotificationParams,
                ...
        }
    }

BIPAcknowledgement ::= SEQUENCE {
        masterEndpointIRI               EndpointIRI,
        slaveEndpointIRI                EndpointIRI,
        linkNumber                      UnsignedInt,
        notificationId                  UnsignedInt,
        params                          CHOICE {
            guiSelectEvent              GUISelectEvent-
        AcknowledgementParams,
            guiStateEvent               GUIStateEvent-
        AcknowledgementParams,
            guiProgressEvent            GUIProgressEvent-
            AcknowledgementParams,
            ...
        },
        returnValue                             BioAPI-RETURN
    }
```

14.2 Предполагается, что вышеуказанные и все остальные определения типа ACH.1 должны находиться в окружении автоматических признаков. Точная спецификация модуля ACH.1 с такими типами приведена в приложении F.

15 Типы БиоАПИ и ПМО БиоАПИ

15.1 Целые числа

15.1.1 В ПМО БиоАПИ определены следующие типы целых чисел АСН.1:

```
UnsignedByte        ::= INTEGER (0..max-unsigned-byte)
UnsignedShort       ::= INTEGER (0..max-unsigned-short)
UnsignedInt         ::= INTEGER (0..max-unsigned-int)
SignedInt           ::= INTEGER (min-signed-int..max-signed-int)
MemoryAddress       ::= INTEGER
```

для которых ссылки значений определены следующим образом:

```
max-unsigned-byte   INTEGER ::= 255
max-unsigned-short  INTEGER ::= 65535
max-unsigned-int    INTEGER ::= 4294967295
min-signed-int      INTEGER ::= -2147483648
max-signed-int      INTEGER ::= 2147483647
```

15.1.2 Встроенное целое число **(INTEGER)** типа АСН.1 всегда появляется с ограниченным диапазоном в корреспонденции с целым типа Си (как правило – **uint8_t)**. Набор значений ограниченного типа АСН.1 совпадает либо является подмножеством набора значений типа Си (может быть одно или более неконвертируемых значений Си). Преобразование между типом АСН.1 и типом Си (в обоих направлениях) должно происходить путем преобразования данных между целым числом абстрактного значения типа АСН.1 и целым числом (со знаком или без знака) значениея Си. Положения, приведенные в разделе 32 настоящего стандарта, применяют в случае, когда сталкиваются с неконвертируемыми целыми значений Си.

15.1.3 Тип АСН.1 **UnsignedByte** появляется в корреспонденции с типом Си **uint8_t**. Набор значений – одинаковый на обоих языках. Преобразование между типом АСН.1 и типом Си (в обоих направлениях) должно выполняться путем преобразования данных между целым абстрактного значения типа АСН.1 и соответствующим 8-битовым целым значением Си без знака.

15.1.4 Тип АСН.1 **UnsignedShort** появляется в корреспонденции с типом Си **uint16_t**. Набор значений одинаков на обоих языках. Преобразование между типом АСН.1 и типом Си (в обоих направлениях)

должно выполняться путем преобразования данных между целым абстрактного значения типа АСН.1 и соответствующим 16-битовым целым значением Си без знака.

15.1.5 Тип АСН.1 **UnsignedInt** появляется в корреспонденции с типом Си **uint32_t**. Набор значений – одинаковый на обоих языках. Преобразование между типом АСН.1 и типом Си (в обоих направлениях) должно выполняться путем преобразования данных между целым абстрактного значения типа АСН.1 и соответствующим 32-битовым целым значением Си без знака.

15.1.6 Тип АСН.1 **SignedInt** появляется в корреспонденции с типом Си **int32_t**. Набор значений одинаков на обоих языках. Преобразование между типом АСН.1 и типом Си (в обоих направлениях) должно выполняться путем преобразования данных между целым абстрактным значением типа АСН.1 и соответствующим 32-битовым целым значением Си без знака.

15.1.7 Тип АСН.1 **MemoryAddress** появляется в корреспонденции с типом Си **void*** или указателем функции типа Си. Преобразование данных между значениями типа АСН.1 **MemoryAddress** и значениями указателя Си в настоящем стандарте не рассматривается.

Примечание – Любое определенное реализацией преобразование данных между значениями **MemoryAddress** и значениями указателя Си является приемлемым только в том случае, пока каждое значение указателя преобразуется в разные целые **MemoryAddress,** и такое целое преобразуется в такое же значение указателя. Тип АСН.1 **MemoryAddress** не встречается в содержимом сообщений ПМО БиоАПИ, обмен которыми происходит между конечными точками ПМО БиоАПИ, в связи с чем его абстрактные значения не кодируются.

15.2 Символы строкового типа

15.2.1 Встроенный тип АСН.1 **UTF8String** появляется в корреспонденции с типом Си **uint8_t***, в котором переменная Си указывает на завершаемый нулем массив октетов, содержащий UTF-8 закодированную строку символов.

15.2.2 Преобразование переменной указателя Си в компонент АСН.1 выполняется следующим образом:

a) если переменная указателя Си имеет значение **NULL,** а компонент АСН.1 – **OPTIONAL**, то компонент АСН.1 отсутствовать;

b) если переменная указателя Си имеет значение **NULL,** а компонент АСН.1 – не **OPTIONAL**, то значение Си не конвертируется и применяют положения раздела 33;

c) если переменная указателя Си имеет значение, отличающееся от **NULL**, то содержимое массива октетов, который выделен переменной Си до первого октета (исключая этот октет) с нулевым значением, должно быть интерпретировано как UTF-8 кодировка строки символов, а компонент АСН.1 должен быть установлен в строку символов.

15.2.3 Преобразование компонента АСН.1 в переменную указателя Си выполняется следующим образом:

a) если компонент АСН.1 **OPTIONAL** отсутствует, то переменной указателя Си принимают **NULL**;

b) если компонент АСН.1 присутствует, то принимают L за значение (в октетах) UTF-8 кодированной строки символов АСН.1; в этом случае новообразованный массив октетов $L + 1$ должен быть заполнен такой UTF-8 кодировкой, следующей после октета с нулевым значением, а переменная Си должна быть установлена в адрес такого массива октетов.

15.3 Унифицированный идентификатор ресурса назначения конечных точек ПМО БиоАПИ

15.3.1 Данный тип АСН.1 в ПМО БиоАПИ определяется следующим образом:

```
EndpointIRI ::= VisibleString (CONSTRAINED BY
        {--The string shall conform to the "absolute-IRI" grammar--
        --defined in IETF RFC 3987--)
```

15.3.2 Тип АСН.1 **EndpointIRI** появляется в корреспонденции с типом Си **uint8_t***, в котором переменная Си указывает на завершаемый нулем массив октетов, содержащий унифицированный идентификатор ресурса.

15.3.3 Преобразование переменной указателя Си в компонент АСН.1 выполняется следующим образом:

a) если переменная указателя Си имеет значение **NULL**, то компонент АСН.1 должен быть установлен в ИИР локальной конечной точки;

b) в противном случае содержание массива октетов, который выделен переменной Си до первого (исключительного) октета с нулевым значением, должно интерпретироваться как код ASCII строки символов, а компонент АСН.1 должен быть установлен в такую строку символов.

15.3.4 Преобразование компонента АСН.1 в переменную указателя Си выполняется следующим образом:

a) если компонент АСН.1 содержит ИИР локальной конечной точки, то переменная указателя Си должна быть переведена в **NULL**;

b) в противном случае, принимают *L* за длину абстрактной строки символов АСН.1; в этом случае новообразованный массив октетов *L* + 1 должен быть заполнен кодом ASCII строки символов АСН.1, следующим после октета с нулевым значением, а переменная Си должна быть установлена в адрес такого массива октетов.

15.4 Тип BioAPI_BFP_LIST_ELEMENT

14.4.1 В ПМО БиоАПИ данный тип Си определяют следующим образом:

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

15.4.2 В ПМО БиоАПИ соответствующий тип АСН.1 определятся следующим образом:

```
BioAPI-BFP-LIST-ELEMENT ::= SEQUENCE {

        category                BioAPI-CATEGORY,
        bfpProductUuid          BioAPI-UUID
}
```

15.4.3 Преобразование между типами АСН.1 и Си (в обоих направлениях) выполняется путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 5.

Таблица 5 – Преобразование данных между членами типа Си **BioAPI_BFP_LIST_ELEMENT** и компонентами АСН.1 типа **BioAPI-BFP-LIST-ELEMENT**

Член типа Си	Компонент типа АСН.1	Номер пункта настоящего стандарта
BFPCategory	Category	15.21
BFPUuid	bfpProductUuid	15.58

Примечание – Член **HostingEndpointIRI** не присутствует в типе Си **BioAPI_BFP_LIST_ELEMENT**. Данный тип Си используется в функции **BioAPI_QueryBFPs**, а все ПБФ в листе ПБФ, возвращаемые указанной функцией, являются такими же главными конечными точками ПМО БиоАПИ, которые являются конечными точками ПМО БиоАПИ, в которых исполняется ПБФ.

15.5 Тип BioAPI_BFP_SCHEMA

15.5.1 Данный тип Си определен в ПМО БиоАПИ следующим образом:

```
typedef struct bioapi_bfp_schema {
        BioAPI_UUID BFPProductUuid;
        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 BFPPropertyUuid;
        BioAPI_DATA BFPProperty;
        uint8_t *HostingEndpointIRI;
    } BioAPI_BFP_SCHEMA;
```

15.5.2 В ПМО БиоАПИ соответствующий тип АСН.1 определятся следующим образом:

```
BioAPI-BFP-SCHEMA ::= SEQUENCE {
        bfpProductUuid      BioAPI-UUID,
        category            BioAPI-CATEGORY,
        description         BioAPI-STRING,
        path                UTF8String,
        specVersion         BioAPI-VERSION,
        productVersion      BioAPI-STRING,
        vendor              BioAPI-STRING,
        supportedFormats    SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                format BioAPI-BIR-BIOMETRIC-DATA-FORMAT,
        factorsMask         BioAPI-BIR-BIOMETRIC-TYPE,
        propertyUuid        BioAPI-UUID,
        property            BioAPI-DATA,
        hostingEndpointIRI  EndpointIRI
    }
```

15.5.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняется путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 6.

Таблица 6 – Преобразование данных между членами типа Си **BioAPI_BFP_SCHEMA** и компонентами ACH.1 типа **BioAPI-BFP-SCHEMA**

Член типа Си	Компонент типа ACH.1	Номер пункта настоящего стандарта
BFPProductUuid	bfpProductUuid	15.58
BFPCategory	Category	15.21
BFPDescription	Description	15.53
Path	Path	15.2
SpecVersion	specVersion	15.59
ProductVersion	productVersion	15.53
Vendor	Vendor	15.53
BFPSupportedFormats, NumSupportedFormats	supportedFormats	15.5.4 и 15.5.5
FactorsMask	factorsMask	15.10
BFPPropertyUuid	propertyUuid	15.58
BFPProperty	Property	15.22
HostingEndpointIRI	hostingEndpointIRI	15.3

15.5.4 Преобразование пары членов Си **BFPSupportedFormats/NumSupportedFormats** в компонент ACH.1 **supportedFormats** выполняют следующим образом: принимают *N* равным значению члена **NumSupportedFormats**; в этом случае каждый первый элемент *N* (типа **BioAPI_BIR_BIOMETRIC_DATA_FORMAT** – см 15.8) в массиве, который выделен членом **BFPSupportedFormats**, должен быть преобразован по порядку в элемент компонента **supportedFormats,** согласно 15.8. У компонента **supportedFormats** должно быть точно *N* элементов.

15.5.5 Преобразование компонента ACH.1 **supportedFormats** в пару членов Си **BFPSupportedFormats/NumSupportedFormats** выполняют следующим образом: принимают *N* равным числу элементов компонента **supportedFormats**; в этом случае новый массив элементов *N* типа **BioAPI_BIR_BIOMETRIC_DATA_FORMAT** (см. 15.8) должен быть заполнен путем преобразования каждого элемента компонента **supportedFormats** в

элемент массива согласно 15.8. Член **BFPSupportedFormats** должен быть установлен в адрес массива, а член **NumSupportedFormats,** должен быть установлен в *N*.

Примечание – Если тип Си **BioAPI_BSP_SCHEMA** (см. 15.19) включает в себя УУИД доступа ПБУ, УУИД доступа ПБФ отсутствует в типе Си **BioAPI_BFP_SCHEMA**. Если ни одна функция БиоАПИ не имеет непосредственного доступа к ПБФ, УУИД доступа ПБФ не требуется. ИИР конечной точки включен в данный тип Си, так как приложению может понадобиться вызов **BioAPIEnumBFPs** для получения списка ПБФ, определения доступности ПБФ в какой-либо конечной точке ПМО БиоАПИ и последующей удаленной загрузки ПБУ в такую конечную точку ПБУ с целью получения возможности использования такого ПБФ.

15.6 Тип BioAPI_BIR

15.6.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct bioapi_bir {
        BioAPI_BIR_HEADER Header;
        BioAPI_DATA BiometricData;
        BioAPI_DATA SecurityBlock;
} BioAPI_BIR;
```

15.6.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR ::= SEQUENCE {
        patronFormatOwner   UnsignedShort,
        patronFormatType    UnsignedShort,
        formattedBIR        OCTET STRING
}
```

15.6.3 Преобразование типа Си в тип АСН.1 выполняется путем преобразования значений типа Си в последовательную форму БиоАПИ ЗБИ (см. ИСО/МЭК 19784-1, приложение В), размещения полученной ЗБИ в октетной строке **formattedBIR** и присвоения **patronFormatOwner** и **patronFormatType** значений 257 и 8 (в указанной последовательности).

Примечание – Для любого абстрактного значения данного типа АСН.1, компонент **formattedBIR** будет состоять из байтов ЗБИ в формате, определенном в

приложении В БиоАПИ. Реализация привязки транспортного протокола может либо включать байтовую строку (как она есть) в кодировку либо преобразовывать ее в другой предшествовавший кодированию формат ведущей организации (см. 13.17).

15.6.4 Преобразование типа АСН.1 в тип Си выполняется путем интерпретирования содержания октетной строки **formattedBIR** как БиоАПИ ЗБИ (см. ИСО/МЭК 19784-1, приложение В) и преобразования его из последовательной формы в значение типа Си при условии, что **patronFormatOwner** и **patronFormatType** представлены значениями 257 и 8 (в указанной последовательности). Если у указанных компонентов будут другие значения, инфраструктура должна создать и отправить сообщение ответа ПМО БиоАПИ, которое соответствует сообщению запроса ПМО БиоАПИ, с возвращаемым значением, установленным в **BioAPIERR_PATRON_FORMAT_NOT_SUPPORTED**.

Примечание – Это происходит в том случае, когда реализация привязки транспортного протокола получает сообщение, содержащее ЗБИ в неподдерживаемом или недопустимом формате, который не может быть преобразован в формат ведущей организации БиоАПИ ПИП (см. 13.18).

15.7 Тип BioAPI_BIR_ARRAY_POPULATION

15.7.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct bioapi_bir_array_population {
        uint32_t NumberOfMembers;
        BioAPI_BIR *Members;
} BioAPI_BIR_ARRAY_POPULATION;
```

15.7.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-ARRAY-POPULATION ::= SEQUENCE {
        members        SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                member BioAPI-BIR
}
```

15.7.3 Преобразование пары Си членов **NumberOfMembers/Members** в компонент **members** АСН.1 выполняется следующим образом: принимают *N*

равным значению члена **NumberOfMembers**, в этом случае каждый из первых элементов *N* (типа **BioAPI_BIR** – см. 15.6) в массиве, который выделен членом **Members**, должен быть преобразован, по порядку, в элемент компонента **members** согласно 15.6. Компонент **members** должен содержать точно *N* элементов.

15.7.4 Преобразование компонента **members** АСН.1 в пару Си членов **NumberOfMembers/Members** выполняется следующим образом: принимают *N*, равным числу элементов компонента **members**; в этом случае новый массив *N* элементов типа **BioAPI_BIR** (см. 15.6) должен быть заполнен путем преобразования каждого элемента компонента **members**, по порядку, в элемент массива согласно в 15.6. Член **Members** должен быть установлен в адрес массива, а член **NumberOfMembers** должен быть установлен в *N*.

15.8 Тип BioAPI_BIR_BIOMETRIC_DATA_FORMAT

15.8.1 В ПМО БиоАПИ данный тип Си определен в следующим образом:

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

15.8.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-BIOMETRIC-DATA-FORMAT ::= SEQUENCE {
        formatOwner UnsignedShort,
        formatType UnsignedShort
    }
```

15.8.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняется путем преобразования индивидуальныч членов Си и компонентов АСН.1 в соответствии с таблицей 7.

Таблица 7 – Преобразование данных между членами типа Си **BioAPI_BIR_BIOMETRIC_DATA_FORMAT** и компонентами АСН.1 типа **BioAPI-BIR-BIOMETRIC-DATA-FORMAT**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
FormatOwner	formatOwner	15.1.4
FormatType	formatType	15.1.4

15.9 Тип BioAPI_BIR_BIOMETRIC_PRODUCT_ID

15.9.1 В ПМО БиоАПИ данный тип Си определятся следующим образом:

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

15.9.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-BIOMETRIC-PRODUCT-ID ::= SEQUENCE {
        productOwner UnsignedShort,
        productType UnsignedShort
    }
```

15.9.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняется путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 8.

Таблица 8 – Преобразование данных между членами типа Си **BioAPI_BIR_BIOMETRIC_PRODUCT_ID** и компонентами АСН.1 типа **BioAPI-BIR-BIOMETRIC-PRODUCT-ID**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
ProductOwner	productOwner	15.1.4
ProductType	productType	15.1.4

15.10 Тип BioAPI_BIR_BIOMETRIC_TYPE

15.10.1 Данный тип Си определятся в ПМО БиоАПИ следующим образом:

```
typedef uint32_t BioAPI_BIR_BIOMETRIC_TYPE;
```

15.10.2 Для поддерживания данного типа Си в ПМО БиоАПИ определены следующие символические константы:

```
#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)
```

15.10.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-BIOMETRIC-TYPE ::= BIT STRING {

    typeMultipleBiometricTypes      (0),
    typeFace                        (1),
    typeVoice                       (2),
    typeFinger                      (3),
    typeIris                        (4),
    typeRetina                      (5),
    typeHandGeometry                (6),
    typeSignatureSign               (7),
```

```
    typeKeystroke                (8),
    typeLipMovement              (9),
    typeGait                     (12),
    typeVein                     (13),
    typeDNA                      (14),
    typeEar                      (15),
    typeFoot                     (16),
    typeScent                    (17),
    typeOther                    (30),
    typePassword                 (31)

} (SIZE(32))
```

15.10.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) совершается следующим образом: Каждый бит битовой строки АСН.1 должен быть преобразован в бит целого Си без знака. Ведущий бит битовой строки (бит 0) должен быть преобразован в наименее значимый бит целого числа без знака (соответствующий значению 0x00000001), а оставшиеся тридцать один бит должны быть преобразованы по порядку.

15.11 Тип BioAPI_BIR_DATA_TYPE

15.11.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_BIR_DATA_TYPE;
```

15.11.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#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)
```

15.11.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-DATA-TYPE ::= SEQUENCE {
        processedLevel      ENUMERATED {
                raw,
```

```
            intermediate,
            processed,
            ...
        },
        flags                   BIT STRING {
            encrypted           (0),
            signed              (1),
            index-present       (3)
        } (SIZE (4))
    }
```

15.11.4 Преобразование между типом Си и типом ACH.1 (в обоих направлениях) выполняется следующим образом:

a) признаки компонента типа ACH.1 должны быть преобразованы в четыре самых больших бита целого числа Си без знака; ведущий бит (бит 0) этого компонента должен быть преобразован в наименьший из четырех самых больших битов целого Си без знака (тот, который соответствует значению 0x10), а оставшиеся три бита должны быть преобразованы по порядку; и

b) компонент **processedLevel** должен быть преобразован в четыре наименьших бита целого Си без знака в соответствии с таблицей 9.

Таблица 9 – Преобразование данных между наименьшими четырьмя битами типа Си **BioAPI_BIR_DATA_TYPE** и компонентом processedLevel типа ACH.1 **BioAPI-BIR-BIOMETRIC-PRODUCT-ID**

Значение в позиции бита 8 0x01	Значение в позиции бита 7 0x02	Значение в позиции бита 6 0x04	Значение в позиции бита 5 0x08	Значение компонента ASN.1 processedLevel
1	0	0	0	Исходное значение
0	1	0	0	Промежуточное значение
0	0	1	0	Обработанное значение
Другие комбинации значений				Отсутствует – значение Си не преобразуется, см. раздел 32

15.12 Тип BioAPI_BIR_HANDLE

15.12.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef int32_t BioAPI_BIR_HANDLE;
```

15.12.2 В ПМО БиоАПИ для поддерживания данного типа Си определены следующие символические константы:

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

15.12.3 ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-HANDLE ::= SignedInt
```

15.12.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняется согласно в 14.1.6.

15.13 Тип BioAPI_BIR_HEADER

15.13.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
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;
```

15.13.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-HEADER ::= SEQUENCE {
```

```
    patronFormatOwner       UnsignedShort,
    patronFormat            Type UnsignedShort,
    formattedBIR            OCTET STRING
}
```

15.13.3 Преобразование типа Си в тип АСН.1 выполняют путем преобразования значения типа Си в последовательную форму БиоАПИ ЗБИ согласно приложению В ИСО/МЭК 19784-1, размещения полученной ЗБИ в компонент **formattedBIR** и установки **patronFormatOwner** и **patronFormatType** в значения 257 и 8 (соответственно).

Примечание – Для любого абстрактного значения данного типа АСН.1, компонент **formattedBIR** состоит из байтов ЗБИ в формате, определенном в приложении В БиоАПИ, с пустыми ББД и SB. Реализация привязки транспортного протокола может либо включать байтовую строку в кодировку без изменений, либо преобразовывать ее в другой предшествовавший кодированию формат ведущей организации (см. 12.17).

15.13.4 Преобразование типа АСН.1 в тип Си совершается путем интерпретирования содержания октетной строки **formattedBIR** как БиоАПИ ЗБИ (см. ИСО/МЭК 19784-1, приложение В) и преобразования его из последовательной формы в значение типа С при условии, что **patronFormatOwner** и **patronFormatType** представлены значениями 257 и 8 (соответственно). Если вышеуказанные компоненты будут иметь другие значения, инфраструктура должна создать и отправить сообщение ответа ПМО БиоАПИ, которое соответствует сообщению запроса ПМО БиоАПИ, с возвращаемым значением, установленным в **BioAPIERR_PATRON_FORMAT_NOT_SUPPORTED**.

Примечание – Это происходит в том случае, когда реализация привязки транспортного протокола получает сообщение, содержащее ЗБИ в неподдерживаемом или недопустимом формате, который не может быть преобразован в формат ведущей организации БиоАПИ ПИП (см. 13.18).

15.14 Тип BioAPI_BIR_PURPOSE

15.14.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_BIR_PURPOSE;
```

15.14.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#define BioAPI_PURPOSE_VERIFY                                   (1)
#define BioAPI_PURPOSE_IDENTIFY                                 (2)
#define BioAPI_PURPOSE_ENROLL                                   (3)
#define BioAPI_PURPOSE_ENROLL_FOR_VEIFICATION_ONLY              (4)
#define BioAPI_PURPOSE_ENROLL_FOR_IDENIFCATION_ONLY             (5)
#define BioAPI_PURPOSE_AUDIT                                    (6)
#define BioAPI_PURPOSE_ANY                                      (7)
```

15.14.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-PURPOSE ::= ENUMERATED {
                Verify,
                identify,
                enroll,
                enrollVerify,
                enrollIdentify,
                audit,
                any,
                ...
        }
```

15.14.4 Преобразование данных между типом Си и типом АСН.1 (в обоих направлениях) выполняется в соответствии с таблицей 10:

Таблица 10 – Преобразование данных между индивидуальными значениями типа Си **BioAPI_BIR_PURPOSE** и абстрактными значениями типа ACH.1 **BioAPI-BIR-PURPOSE**

Значение типа Си	Абстрактное значение типа ACH.1
1 (BioAPI_PURPOSE_VERIFY)	verify
2 (BioAPI_PURPOSE_IDENTIFY)	identify
3 (BioAPI_PURPOSE_ENROLL)	enroll
4 (BioAPI_PURPOSE_ENROLL_FOR_VEIFICATION_ONLY)	enrollVerify
5 (BioAPI_PURPOSE_ENROLL_FOR_IDENIFCATION_ONLY)	audit enrollIdentify
6 (BioAPI_PURPOSE_AUDIT)	audit
7 (BioAPI_PURPOSE_ANY)	any
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.15 Тип BioAPI_BIR_SECURITY_BLOCK_FORMAT

15.15.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

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

15.15.2 В ПМО БиоАПИ соответствующий тип ACH.1 определен следующим образом:

```
BioAPI-BIR-SECURITY-BLOCK-FORMAT ::= SEQUENCE {
        formatOwner         UnsignedShort,
        formatType          UnsignedShort
    }
```

15.15.3 Преобразование данных между типом Си и типом ACH.1 (в обоих направлениях) выполняют путем преобразования между

индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 11.

Таблица 11 – Преобразование данных между членами типа Си **BioAPI_BIR_SECURITY_BLOCK_FORMAT** и компонентами типа АСН.1 **BioAPI-BIR-SECURITY-BLOCK-FORMAT**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
SecurityFormatOwner	formatOwner	15.1.4
SecurityFormatType	formatType	15.1.4

15.16 Тип BioAPI_BIR_SUBTYPE

15.16.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_BIR_SUBTYPE;
```

15.16.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#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)
```

15.16.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-SUBTYPE ::= CHOICE {
        any-subtype BIT STRING {
                left                    (0),
                right                   (1),
                thumb                   (2),
                pointerFinger           (3),
                middleFinger            (4),
                ringFinger              (5),
                littleFinger            (6)} (SIZE(7)),
        vein-only-subtype BIT STRING {
                left                    (0),
                right                   (1),
                veinPalm                (2),
                veinBackofhand          (3),
                veinWrist               (4)} (SIZE(7))
}
```

15.16.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют следующим образом: каждый бит битовой строки АСН.1 должен быть преобразован в бит целого Си без знака. Ведущий бит битовой строки (бит 0) должен быть преобразован в наименьший бит целого числа без знака (соответствующий значению 0x01), а оставшиеся семь битов должны быть преобразованы по порядку.

15.17 Тип BioAPI_BIR_SUBTYPE_MASK

15.17.1 В ПМО БиоАПИ данный тип Си определенследующим образом:

```
typedef uint32_t BioAPI_BIR_SUBTYPE_MASK;
```

15.17.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#define BioAPI_BIR_SUBTYPE_LEFT_BIT                 (0x00000001)
#define BioAPI_BIR_SUBTYPE_RIGHT_BIT                (0x00000002)
#define BioAPI_BIR_SUBTYPE_LEFT_THUMB_BIT           (0x00000004)
#define BioAPI_BIR_SUBTYPE_LEFT_POINTERFINGER_BIT   (0x00000008)
#define BioAPI_BIR_SUBTYPE_LEFT_MIDDLEFINGER_BIT (0x00000010)
```

```
#define BioAPI_BIR_SUBTYPE_LEFT_RINGFINGER_BIT          (0x00000020)
#define BioAPI_BIR_SUBTYPE_LEFT_LITTLEFINGER_BIT        (0x00000040)
#define BioAPI_BIR_SUBTYPE_RIGHT_THUMB_BIT              (0x00000080)
#define BioAPI_BIR_SUBTYPE_RIGHT_POINTERFINGER_BIT      (0x00000100)
#define BioAPI_BIR_SUBTYPE_RIGHT_MIDDLEFINGER_BIT       (0x00000200)
#define BioAPI_BIR_SUBTYPE_RIGHT_RINGFINGER_BIT         (0x00000400)
#define BioAPI_BIR_SUBTYPE_RIGHT_LITTLEFINGER_BIT       (0x00000800)
#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)
```

15.17.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BIR-SUBTYPE-MASK ::= BIT STRING {
        left                    (0),
        right                   (1),
        left-thumb              (2),
        left-pointerfinger      (3),
        left-middlefinger       (4),
        left-ringfinger         (5),
        left-littlefinger       (6),
        right-thumb             (7),
        right-pointerfinger     (8),
        right-middlefinger      (9),
        right-ringfinger        (10),
        right-littlefinger      (11),
        left-vein-palm          (12),
        left-vein-backofhand    (13),
        left-vein-wrist         (14),
        right-vein-palm         (15),
        right-vein-backofhand   (16),
        right-vein-wrist        (17)
    } (SIZE(32))
```

15.17.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют следующим образом: каждый бит битовой строки АСН.1 должен быть преобразован в бит целого Си без знака. Ведущий бит битовой строки (бит 0) должен быть преобразован в наименьший бит целого

числа без знака (соответствующий значению 0x00000001), а оставшиеся тридцать один бит должны быть преобразованы по порядку.

15.18 Тип BioAPI_BOOL

15.18.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef int8_t BioAPI_BOOL;
```

15.18.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

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

15.18.3 Соответствующим типом АСН.1 в ПМО БиоАПИ является встроенный тип **BOOLEAN**.

15.18.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 12:

Таблица 12 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_BOOL** и абстрактными значениями АСН.1 типа **BOOLEAN**

Значение типа Си	Абстрактное значение типа АСН.1
0 (BioAPI_FALSE)	FALSE
1 (BioAPI_TRUE)	TRUE
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.19 Тип BioAPI_BSP_SCHEMA

15.19.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct _bioapi_bsp_schema {
        BioAPI_UUID BSPProductUuid;
        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;
        uint8_t *HostingEndpointIRI;
        BioAPI_UUID BSPAccessUuid;
    } BioAPI_BSP_SCHEMA;
```

15.19.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-BSP-SCHEMA ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID,
        description             BioAPI-STRING,
        path                    UTF8String,
        specVersion             BioAPI-VERSION,
        productVersion          BioAPI-STRING,
        vendor                  BioAPI-STRING,
        supportedFormats        SEQUENCE (SIZE(0..max-unsigned-int)) OF
```

```
                                    format BioAPI-BIR-BIOMETRIC-DATA-FORMAT,
                                    factorsMask BioAPI-BIR-BIOMETRIC-
                                    TYPE,
    operations                      BioAPI-OPERATIONS-MASK,
    options                         BioAPI-OPTIONS-MASK,
    payloadPolicy                   BioAPI-FMR,
    maxPayloadSize                  UnsignedInt,
    defaultVerifyTimeout            SignedInt,
    defaultIdentifyTimeout          SignedInt,
    defaultCaptureTimeout           SignedInt,
    defaultEnrollTimeout            SignedInt,
    defaultCalibrateTimeout         SignedInt,
    maxBSPDbSize                    UnsignedInt,
    maxIdentify                     UnsignedInt,
    hostingEndpointIRI              EndpointIRI,
    bspAccessUuid                   BioAPI-UUID
}
```

15.19.3 Преобразование данных между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 13.

Таблица 13 – Преобразование данных между членами типа Си **BioAPI_BSP_SCHEMA** и компонентами ACH.1 типа **BioAPI-BSP-SCHEMA**

Член типа Си	Компонент типа ACH.1	Пункт настоящего стандарта
BSPProductUuid	bspProductUuid	15.58
BSPDescription	description	15.53
Path	Path	15.2
SpecVersion	specVersion	15.59
ProductVersion	productVersion	15.53
Vendor	vendor	15.53
BSPSupportedFormats, NumSupportedFormats	supportedFormats	15.19.4 и 15.19.5
FactorsMask	factorsMask	15.10
Operations	operations	15.48
Options	options	15.49
PayloadPolicy	payloadPolicy	15.32
MaxPayloadSize	maxPayloadSize	15.1.5
DefaultVerifyTimeout	defaultVerifyTimeout	15.1.6
DefaultIdentifyTimeout	defaultIdentifyTimeout	15.1.6
DefaultCaptureTimeout	defaultCaptureTimeout	15.1.6
DefaultEnrollTimeout	defaultEnrollTimeout	15.1.6
DefaultCalibrateTimeout	defaultCalibrateTimeout	15.1.6
MaxBSPDbSize	maxBSPDbSize	15.1.5
MaxIdentify	maxIdentify	15.1.5
HostingEndpointIRI	hostingEndpointIRI	15.3
BSPAccessUuid	bspAccessUuid	15.58

15.19.4 Преобразование пары Си членов **BSPSupportedFormats/NumSupportedFormats** в компонент АСН.1 **supportedFormats** выполняют следующим образом: принимают *N* за равным значению члена **NumSupportedFormats**; в этом случае каждый из первых элементов *N* (типа **BioAPI_BIR_BIOMETRIC_DATA_FORMAT** – см. 14.8) в массиве, который выделен членом **BSPSupportedFormats** должен быть преобразован по порядку в элемент компонента **supportedFormats** согласно в 14.8. Компонент **supportedFormats** должен содержать точное число *N* элементов.

15.19.5 Преобразование компонента АСН.1 **supportedFormats** в пару Си членов **BSPSupportedFormats/NumSupportedFormats** выполняют следующим образом: принимают *N* равным числу элементов компонента **supportedFormats**; в этом случае новый массив элементов *N* типа **BioAPI_BIR_BIOMETRIC_DATA_FORMAT** (см. 14.8) должен быть заполнен путем преобразования каждого элемента компонента **supportedFormats** по порядку в элемент массива, согласно 14.8. Член **BSPSupportedFormats** должен быть установлен в адрес множества, а член **NumSupportedFormats** быть установлен в *N*.

15.20 Тип BioAPI_CANDIDATE

15.20.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

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

15.20.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-CANDIDATE ::= SEQUENCE {
        bir                                     CHOICE {
```

```
                birInDatabase          BioAPI-UUID,
                birInArray             UnsignedInt,
                birInPresetArray       UnsignedInt
            },
            fmrAchieved BioAPI-FMR
        }
```

15.20.3 Преобразование между типом Си и типом ACH.1 (в обоих направлениях) выполняется путем преобразования между индивидуальными членами Си и компонентами ACH.1 в соответствии с таблицей 14.

Таблица 14 – Преобразование данных между членами типа Си **BioAPI_CANDIDATE** и компонентами ACH.1 типа **BioAPI-CANDIDATE**

Член типа Си	Компонент типа ACH.1	Пункт настоящего стандарта
Type, BIR	Bir	15.20.4 и 15.20.5
FMRAchieved	fmr Achieved	15.32

15.20.4 Преобразование пары членов Си **Type/BIR** в компонент ACH.1 **bir** выполняют в соответствии с таблицей 15 с выполнением следующих действий (в указанном порядке):

a) определение наличия альтернативного компонента **BIR**, который основан на значении **Type**;

b) выбор альтернативы компонента **bir**, который соответствует значению **Type**;

c) преобразование члена Си в альтернативу компонента **bir**.

15.20.5 Преобразование компонента ACH.1 **bir** в пару компонентов членов Си **Type/BIR** выполняют в соответствии с таблицей 15 с выполнением следующих действий (в указанном порядке):

a) установка члена **Type,** который основывается на имеющейся альтернативе компонента **bir**;

b) выбор соответствующей альтернативы члена **BIR**;

c) преобразование компонента ACH.1 в член Си.

Таблица 15 – Преобразование данных между альтернативами члена **BIR** типа Си **BioAPI_CANDIDATE** и альтернативами компонента АСН.1 **bir** типа **BioAPI-CANDIDATE**

Значение члена Type	Альтернатива члена BIR	Альтернатива члена bir	Пункт настоящего стандарта
1 (BioAPI_DB_TYPE)	BIRInDataBase	birInDatabase	Раздел 19 и 15.58
2 (BioAPI_ARRAY_TYPE)	BIRInArray	birInArray	Раздел 19 и 15.1.5
3 (BioAPI_PRESET_ARRAY_TYPE)	BIRInPresetArray	birInPresetArray	Раздел 19 и 15.1.5
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33		

15.21 Тип BioAPI_CATEGORY

15.21.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_CATEGORY;
```

15.21.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

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

15.21.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-CATEGORY ::= ENUMERATED {
                archive,
                comparisonAlgorithm,
                processingAlgorithm,
                sensor,
                ...
        }
```

15.21.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 16.

Таблица 16 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_CATEGORY** и абстрактными значениями АСН.1 типа **BioAPI-CATEGORY**

Значение типа Си	Абстрактное значение типа АСН.1
1 (BioAPI_CATEGORY_ARCHIVE)	Archive
2(BioAPI_CATEGORY_COMPARISON_ALG)	comparisonAlgorithm
4(BioAPI_CATEGORY_PROCESSING_ALG)	processingAlgorithm
8 (BioAPI_CATEGORY_SENSOR)	Sensor
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.22 Тип BioAPI_DATA

15.22.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

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

15.22.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-DATA ::= OCTET STRING (SIZE(0..max-unsigned-int))
```

15.22.3 Преобразование переменной указателя Си в компонент АСН.1 выполняют следующим образом:

a) если компонент **Data** указателя Си имеет значение **NULL,** а компонент АСН.1 – **OPTIONAL**, компонент АСН.1 должен отсутствовать;

b) если компонент **Data** указателя Си имеет значение **NULL,** а компонент АСН.1 не **OPTIONAL**, значение Си не конвертируется и применяют раздел 33;

c) если компонент **Data** указателя Си имеет значение отличающееся от **NULL**, то принимают *L* равным значению члена **Length** типа Си; в этом случае первые октеты *L* массива октетов, которые выделены членом **Data** типа Си, должны образовывать октетную строку, которая должна быть назначена в абстрактное значение АСН.1.

15.22.4 Преобразование компонента АСН.1 в переменную указателя С выполняютя следующим образом:

a) если компонент АСН.1 **OPTIONAL** отсутствует, член **Data** переменной Си должен быть установлен на **NULL**, а член **Length** переменной Си должен быть установлен на 0;

b) если компонент АСН.1 присутствует, принимают *L* равный длине абстрактной октетной строки АСН.1; в этом случае новый массив октетов *L* должен быть заполнен октетами такой октетной строки; член **Data** переменной Си должен быть установлен в адрес такой октетной строки, а член **Length** переменной Си должен быть равен *L*.

15.23 Тип BioAPI_DATE

15.23.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct bioapi_date {
        uint16_t Year;
        uint8_t Month;
        uint8_t Day;
    } BioAPI_DATE;
```

15.23.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-DATE ::= SEQUENCE {
        year        INTEGER (0 | 1900..9999),
        month       INTEGER (0..12),
        day         INTEGER (0..31)
    }
```

15.23.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняется путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 17.

Таблица 17 – Преобразование данных между членами типа Си **BioAPI_DATE** и компонентами АСН.1 типа **BioAPI-DATE**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
Year	year	15.1.2
Month	month	15.1.2
Day	day	15.1.2

15.23.4 Абстрактное значение типа **BioAPI-DATE**, который состоит из трех компонентов **year**, **month** and **day**, имеющих значение 0, допускается. В отличие от указанного определенного абстрактного значения, все значения **year,** не входящие в диапазон от 1900 до 9999, все значения **month,** не входящие в диапазон от 1 до 12, и все значения **day,** не входящие в диапазон от 1 до 31 не могут быть преобразованы, при обнаружении таких значений выполняют действие, указанные в разделе 33.

15.24 Тип BioAPI_DB_ACCESS_TYPE

15.24.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_DB_ACCESS_TYPE;
```

15.24.2 В ПМО БиоАПИ для поддерживания данного типа Си определены следующие символические константы:

```
#define BioAPI_DB_ACCESS_READ        (0x00000001)
#define BioAPI_DB_ACCESS_WRITE       (0x00000002)
```

15.24.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-DB-ACCESS-TYPE ::= BIT STRING {
            read        (0),
            write       (1)
    } (SIZE(32))
```

15.24.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют следующим образом: Каждый бит битовой строки

АСН.1 должен быть преобразован в бит целого значения Си без знака. Ведущий бит битовой строки (бит 0) должен быть преобразован в наименьший бит целого числа без знака (соответствующий значению 0x00000001), а оставшиеся тридцать один бит должны быть преобразованы по порядку.

15.25 Тип BioAPI_DB_MARKER_HANDLE

15.25.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_DB_MARKER_HANDLE;
```

15.25.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-DB-MARKER-HANDLE ::= UnsignedInt
```

15.25.3 Преобразования между типом Си и типом АСН.1 (в обоих направлениях) выполняют согласно 15.1.5.

15.26 Тип BioAPI_DB_HANDLE

15.26.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef int32_t BioAPI_DB_HANDLE;
```

15.2.6.2 В ПМО БиоАПИ для поддерживания данного типа Си определены следующие символические константы:

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

15.26.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-DB-HANDLE ::= SignedInt
```

15.26.4 Преобразования между типом Си и типом АСН.1 (в обоих направлениях) выполняют согласно 15.1.6.

15.27 Тип BioAPI_DBBIR_ID

15.27.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

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

15.27.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-DBBIR-ID ::= SEQUENCE {
        dbHandle        BioAPI-DB-HANDLE,
        keyValue        BioAPI-UUID
    }
```

15.27.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 18.

Таблица 18 – Преобразование данных между членами типа Си **BioAPI_DBBIR_ID** и компонентами АСН.1 типа **BioAPI-DBBIR-ID**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
DbHandle	dbHandle	15.26
KeyValue	keyValue	15.58

15.28 Тип BioAPI_DTG

15.28.1 В ПМО БиоАПИ данный тип Си определяют следующим образом:

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

15.28.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-DTG ::= SEQUENCE {
        date    BioAPI-DATE,
        time    BioAPI-TIME
    }
```

15.28.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 19.

Таблица 19 – Преобразование данных между членами типа Си **BioAPI_DTG** и компонентами АСН.1 типа **BioAPI-DTG**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
Date	date	15.23
Time	time	15.54

15.29 Тип BioAPI_ERROR_INFO

15.29.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

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

15.29.2 В ПМО БиоАПИ отсутствует соответствующий тип АСН.1.

Примечание – Данный тип Си используется только для функции **BioAPI_GetLastErrorInfo** (см. 16.56), для которой нет соответствующего типа сообщений ПМО БиоАПИ.

15.30 Тип BioAPI_EVENT

15.30.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_EVENT;
```

15.30.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#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)
```

15.30.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-UNIT-EVENT-TYPE ::= ENUMERATED {
        insert,
        remove,
        fault,
        sourcePresent,
        sourceRemoved,
        ...
    }
```

15.30.4. Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 20.

Таблица 20 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_EVENT** и абстрактными значениями АСН.1 типа **BioAPI-UNIT-EVENT-TYPE**

Значение типа Си	Абстрактное значение типа АСН.1
1 (BioAPI_NOTIFY_INSERT)	insert
2 (BioAPI_NOTIFY_REMOVE)	remove
3 (BioAPI_NOTIFY_FAULT)	fault
4 BioAPI_NOTIFY_SOURCE_PRESENT)	sourcePresent
5 BioAPI_NOTIFY_SOURCE_REMOVED)	sourceRemoved
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 32

15.31 Тип BioAPI_EVENT_MASK

15.31.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_EVENT_MASK;
```

15.31.2 В БиоАПИ для поддерживания данного типа Си определены следующие символические константы:

```
#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)
```

15.31.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-UNIT-EVENT-TYPE-MASK ::= BIT STRING {
        insert          (0),
        remove          (1),
        fault           (2),
        sourcePresent   (3),
        sourceRemoved   (4)
    } (SIZE(32))
```

15.31.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют следующим образом: каждый бит битовой строки АСН.1 должен быть преобразован в бит целого значения Си без знака. Ведущий бит битовой строки (бит 0) должен быть преобразован в наименьший бит целого числа без знака, соответствующий значению 0x00000001, а оставшиеся тридцать один бит должны быть преобразованы по порядку.

15.32 Тип BioAPI_FMR

15.32.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef int32_t BioAPI_FMR;
```

15.32.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-FMR ::= SignedInt
```

15.32.3 Преобразования между типом Си и типом АСН.1 (в обоих направлениях) выполняют согласно 15.1.6.

15.33 Тип BioAPI_FRAMEWORK_SCHEMA

15.33.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct bioapi_framework_schema {
        BioAPI_UUID FwProductUuid;
        BioAPI_STRING FwDescription;
        uint8_t *Path;
```

```
        BioAPI_VERSION SpecVersion;
        BioAPI_STRING ProductVersion;
        BioAPI_STRING Vendor;
        BioAPI_UUID FwPropertyUuid;
        BioAPI_DATA FwProperty;
        uint8_t *HostingEndpointIRI;
} BioAPI_FRAMEWORK_SCHEMA;
```

15.33.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-FRAMEWORK-SCHEMA ::= SEQUENCE {
        fwProductUuid           BioAPI-UUID,
        description             BioAPI-STRING,
        path                    UTF8String,
        specVersion             BioAPI-VERSION,
        productVersion          BioAPI-STRING,
        vendor                  BioAPI-STRING,
        propertyUuid            BioAPI-UUID,
        property                BioAPI-DATA,
        hostingEndpointIRI      EndpointIRI
}
```

15.33.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 21.

Таблица 21 – Преобразование данных между членами типа Си **BioAPI_FRAMEWORK_SCHEMA** и компонентами АСН.1 типа **BioAPI-FRAMEWORK-SCHEMA**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
FwProductUuid	fwProductUuid	15.58
FwDescription	description	15.53
Path	path	15.2
SpecVersion	specVersion	15.59
ProductVersion	productVersion	15.53
Vendor	vendor	15.53
FwPropertyUuid	propertyUuid	15.58
FwProperty	property	15.22
HostingEndpointIRI	hostingEndpointIRI	15.3

15.34 Тип BioAPI_GUI_BITMAP

15.34.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct bioapi_gui_bitmap {
        BioAPI_BIR_SUBTYPE_MASK SubtypeMask;
        uint32_t Width;
        uint32_t Height;
        BioAPI_DATA Bitmap;
} BioAPI_GUI_BITMAP;
```

15.34.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-GUI-BITMAP ::= SEQUENCE {
        subtypeMask     BioAPI-BIR-SUBTYPE-MASK,
        width           UnsignedInt,
        height          UnsignedInt,
        bitmap          BioAPI-DATA OPTIONAL
}
```

15.34.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 22.

Таблица 22 – Преобразование данных между членами типа Си **BioAPI_GUI_BITMAP** и компонентами АСН.1 типа **BioAPI-GUI-BITMAP**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
SubtypeMask	subtypemask	15.17
Width	width	15.1.5
Height	height	15.1.5
Bitmap	bitmap	15.22

15.35 Тип BioAPI_GUI_BITMAP_ARRAY

15.35.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct bioapi_gui_bitmap_array {
        uint32_t NumberOfMembers;
        BioAPI_GUI_BITMAP *GuiBitmaps;
    } BioAPI_GUI_BITMAP_ARRAY;
```

15.35.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-GUI-BITMAP-ARRAY ::= SEQUENCE {
        guiBitmaps          SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                    guiBitmap BioAPI-GUI-BITMAP
    }
```

15.35.3 Преобразование пары членов Си **NumberOfMembers**/**Members** в компонент АСН.1 **members** выполняют следующим образом: принимают *N* равным значению члена **NumberOfMembers**, в этом случае каждый из первых элементов *N* (типа **BioAPI_GUI_BITMAP** – см. 15.34) в массиве, выделенном членом **Members**, должен быть преобразован по порядку в элемент

компонента **members** согласно 15.34. У компонента **members** должно быть точное число элементов *N*.

15.35.4 Преобразование компонента АСН.1 **members** в пару членов Си **NumberOfMembers/Members** выполняют следующим образом: принимают *N* равным числу элементов компонента **members**, в этом случае новый массив элементов *N* типа **BioAPI_GUI_BITMAP** (см. 15.34) должен быть заполнен путем преобразования каждого элемента компонента **members** по порядку в элемент массива согласно 15.34. Член **Members** должен быть установлен в адрес массива, а **NumberOfMembers** должен быть установлен на *N*.

15.36 Тип BioAPI_GUI_EVENT_SUBSCRIPTION

15.36.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
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;
```

15.36.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-GUI-EVENT-SUBSCRIPTION ::= SEQUENCE {
                subscriberEndpointIRI           EndpointIRI,
                guiEventSubscriptionUuid        BioAPI-UUID,
                guiSelectEventSubscribed        BOOLEAN,
                guiStateEventSubscribed         BOOLEAN,
                guiProgressEventSubscribed      BOOLEAN
        }
```

15.36.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 23.

Таблица 23 – Преобразование данных между членами типа Си **BioAPI_GUI_EVENT_SUBSCRIPTION** и компонентами типа АСН.1 **BioAPI-GUI-EVENT-SUBSCRIPTION**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
SubscriberEndpointIRI	subscriberEndpointIRI	15.3
GUIEventSubscriptionUuid	guiEventSubscriptionUuid	15.58
GUISelectEventSubscribed	guiSelectEventSubscribed	15.18
GUIStateEventSubscribed	guiStateEventSubscribed	15.18
GUIProgressEventSubscribed	guiProgressEventSubscribed	15.18

15.37 Тип BioAPI_GUI_MOMENT

15.37.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_GUI_MOMENT;
```

15.37.2 В БиоАПИ для поддерживания данного типа Си определены следующие символические константы:

```
#define BioAPI_GUI_MOMENT_BEFORE_START (1)
#define BioAPI_GUI_MOMENT_DURING (2)
#define BioAPI_GUI_MOMENT_AFTER_END (3)
```

15.37.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-GUI-MOMENT ::= ENUMERATED {
        beforeStart,
        during,
        afterEnd,
        ...
}
```

15.37.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 24.

Таблица 24 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_GUI_MOMENT** и абстрактными значениями АСН.1 типа **BioAPI-GUI-MOMENT**

Значение типа Си	Абстрактное значение типа АСН.1
1(BioAPI_GUI_MOMENT_BEFORE_START)	before-start
2 (BioAPI_GUI_MOMENT_DURING)	during
3 (BioAPI_GUI_MOMENT_AFTER_END)	after-end
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.38 Тип BioAPI_GUI_ENROLL_TYPE

15.38.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_GUI_ENROLL_TYPE;
```

15.38.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#define BioAPI_GUI_ENROLL_TYPE_TEST_VERIFY          (0x00000001)
#define BioAPI_GUI_ENROLL_TYPE_MULTIPLE_CAPTURE     (0x00000002)
```

15.38.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-GUI-ENROLL-TYPE ::= BIT STRING {
                        testVerify (0),
                        multipleCapture (1)
    } (SIZE(32))
```

15.38.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют следующим образом: каждый бит битовой строки АСН.1 должен быть преобразован в бит целого числа Си без знака. Наибольший бит битовой строки (бит 0) должен быть преобразован в наименее

значимый бит целого числа без знака, соответствующий значению 0x00000001, а оставшиеся тридцать один бит должны быть преобразованы по порядку.

15.39 Тип BioAPI_GUI_OPERATION

15.39.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_GUI_OPERATION;
```

15.39.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#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)
```

15.39.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-GUI-OPERATION ::= ENUMERATED {
        capture,
        process,
        createtemplate,
        Verifymatch,
        identifymatch,
        Verify,
        identify,
        enroll,
        ...
    }
```

15.39.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 25.

Таблица 25 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_GUI**_OPERATION и абстрактными значениями ACH.1 типа **BioAPI-GUI-OPERATION**

Значение типа Си	Абстрактное значение типа ACH.1
1 (BioAPI_GUI_OPERATION_CAPTURE)	capture
2 (BioAPI_GUI_OPERATION_PROCESS)	process
3(BioAPI_GUI_OPERATION_CREATETEMPLATE)	createtemplate
4 (BioAPI_GUI_OPERATION_VERIFYMATCH)	verifymatch
5 (BioAPI_GUI_OPERATION_IDENTIFYMATCH)	identifymatch
6 (BioAPI_GUI_OPERATION_VERIFY)	verify
7 (BioAPI_GUI_OPERATION_IDENTIFY)	identify
8 (BioAPI_GUI_OPERATION_ENROLL)	enroll
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.40 Тип BioAPI_GUI_RESPONSE

15.40.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_GUI_RESPONSE;
```

15.40.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#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)
```

15.40.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-GUI-RESPONSE ::= ENUMERATED {
                default,
                opComplete,
                opCancel,
                cycleStart,
                cycleRestart,
                subopStart,
                subopNext,
                progressContinue,
                progressCancel,
                recapture,
                ...
        }
```

15.40.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 26.

Таблица 26 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_GUI_RESPONSE** и абстрактными значениями АСН.1 типа **BioAPI-GUI-RESPONSE**

Значение типа Си	Абстрактное значение типа АСН.1
0 (BioAPI_GUI_RESPONSE_DEFAULT)	default
1 (BioAPI_GUI_RESPONSE_OP_COMPLETE)	opComplete
2 (BioAPI_GUI_RESPONSE_OP_CANCEL)	opCancel
3 (BioAPI_GUI_RESPONSE_CYCLE_START)	cycleStart
4 (BioAPI_GUI_RESPONSE_CYCLE_RESTART)	cycleRestart
5 (BioAPI_GUI_RESPONSE_SUBOP_START)	subopStart
6 (BioAPI_GUI_RESPONSE_SUBOP_NEXT)	subopNext
7 (BioAPI_GUI_RESPONSE_PROGRESS_CONTINUE)	progressContinue
8 (BioAPI_GUI_RESPONSE_PROGRESS_ABORT)	progressAbort
9 (BioAPI_GUI_RESPONSE_RECAPTURE)	recapture
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.41 Тип BioAPI_GUI_SUBOPERATION

15.41.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_GUI_SUBOPERATION;
```

15.41.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#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)
```

15.41.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-GUI-SUBOPERATION ::= ENUMERATED {
        capture,
        process,
        createtemplate,
        Verifymatch,
        identifymatch,
        ...
    }
```

15.41.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 27.

Таблица 27 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_GUI_SUBOPERATION** и абстрактными значениями АСН.1 типа **BioAPI-GUI-SUBOPERATION**

Значение типа Си	Абстрактное значение типа АСН.1
1 (BioAPI_GUI_SUBOPERATION_CAPTURE)	capture
2 (BioAPI_GUI_SUBOPERATION_PROCESS)	process
3(BioAPI_GUI_SUBOPERATION_CREATETEMPLATE)	createtemplate
4 (BioAPI_GUI_SUBOPERATION_VERIFYMATCH)	verifymatch
5 (BioAPI_GUI_SUBOPERATION_IDENTIFYMATCH)	identifymatch
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.42 Тип BioAPI_HANDLE

15.42.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_HANDLE;
```

15.42.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-HANDLE ::= UnsignedInt
```

15.42.3 Преобразования между типом Си и типом АСН.1 (в обоих направлениях) выполняют согласно 15.1.5.

15.43 Тип BioAPI_IDENTIFY_POPULATION

15.43.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

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

15.43.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-IDENTIFY-POPULATION ::= SEQUENCE {
        birs                    CHOICE {
                birDataBase     BioAPI-DB-HANDLE,
                birArray        BioAPI-BIR-ARRAY-POPULATION,
                birPresetArray      NULL
        }
}
```

15.43.3 Преобразование пары членов Си **Type/BIRs** в компонент АСН.1 **birs** выполняют в соответствии с таблицей 28, при этом выполняют следующие действия в указанном порядке:

a) определяют наличие альтернативного компонента **BIRs**, который основывается на значении **Type Type**;

b) выбирают альтернативу компонента **birs**, который соответствует значению **Type**;

c) преобразовывают член Си в альтернативу компонента **birs**.

15.43.4 Преобразование компонента АСН.1 **birs** в пару компонентов членов Си **Type/BIRs** выполняют в соответствии с таблицей 28, при этом выполняют следующие действия в указанном порядке:

a) установка члена **Type,** который основывается на имеющейся альтернативе компонента **birs**;

b) выбирают соответствующую альтернативу члена **BIRs**;

c) преобразовывают компонент АСН.1 в член Си.

Таблица 28 – Преобразование данных между альтернативами члена **BIR** типа Си **BioAPI_IDENTIFY_POPULATION** и альтернативами компонента АСН.1 **bir** типа **BioAPI-CANDIDATE**

Значение члена Type	Альтернатива члена BIRs	Альтернатива члена birs	Пункт настоящего стандарта
1 (BioAPI_DB_TYPE)	BIRDataBase	birDataBase	Раздел 19 и 15.26
2 (BioAPI_ARRAY_TYPE)	BIRArray	birArray	Раздел 19 и 15.7
3(BioAPI_PRESET_ARRAY_TYPE)	BIRArray	birPresetArray	15.43.5 и 15.43.6
Другие значения	Отсутствует – значение Си не преобразуется, см. разд. 33		

15.43.5 Преобразование члена Си **BIRArray** в компонент АСН.1 **birPresetArray** выполняют путем установки компонента **birPresetArray** на **NULL**. Любое значение **BIRArray** отличное от **NULL** не преобразовывается, в случае обнаружения таких значений применяется раздел 33.

15.43.6 Преобразование компонента АСН.1 **birPresetArray** в член Си **BIRArray** выполняется путем установки члена **BIRArray** в **NULL**.

15.44 Тип BioAPI_IDENTIFY_POPULATION_TYPE

15.44.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_IDENTIFY_POPULATION_TYPE;
```

15.44.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#define BioAPI_DB_TYPE (1)
#define BioAPI_ARRAY_TYPE (2)
#define BioAPI_PRESET_ARRAY_TYPE (3)
```

15.44.3 В ПМО БиоАПИ отсутствует соответствующий тип АСН.1

Примечание – Данный тип Си используется в типах Си **BioAPI_CANDIDATE** (см. 15.20) и **BioAPI_IDENTIFY_POPULATION** (см. 15.43) и

определяет выбор одного из трех способов определения популяции ЗБИ в операции идентификации. В соответствующих типах АСН.1 **BioAPI-CANDIDATE** и **BioAPI-IDENTIFY-POPULATION** выбор одного из трех способов определения популяции ЗБИ представлен конструкцией **CHOICE**.

15.45 Тип BioAPI_INDICATOR_STATUS

15.45.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_INDICATOR_STATUS;
```

15.45.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#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)
```

15.45.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-INDICATOR-STATUS ::= ENUMERATED {
        accept,
        reject,
        ready,
        busy,
        failure,
        ...
    }
```

15.45.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 29.

Таблица 29 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_INDICATOR_STATUS** и абстрактными значениями АСН.1 типа **BioAPI-INDICATOR-STATUS**

Значение типа Си	Абстрактное значение типа АСН.1
1 (BioAPI_INDICATOR_ACCEPT)	accept
2 (BioAPI_INDICATOR_REJECT)	reject
3 (BioAPI_INDICATOR_READY)	ready
4 (BioAPI_INDICATOR_BUSY)	busy
5 (BioAPI_INDICATOR_FAILURE)	failure
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.46 Тип BioAPI_INPUT_BIR

15.46.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
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;
```

15.46.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-INPUT-BIR ::= SEQUENCE {
        inputBIR                CHOICE {
            birInDB             BioAPI-DBBIR-ID,
            birInBSP            BioAPI-BIR-HANDLE,
            bir                 BioAPI-BIR
        }
    }
```

15.46.3 Преобразование пары членов Си **Form/InputBIR** в компонент АСН.1 **inputBIR** выполняют в соответствии с таблицей 30, при этом выполняют следующие действия в указанном порядке:

a) определение наличия альтернативного компонента **InputBIR**, который основывается на значении **Form**;

b) выбор альтернативы компонента **inputBIR,** который соответствует значению **Form**;

c) преобразование члена Си в альтернативу компонента **inputBIR**.

15.46.4 Преобразование компонента АСН.1 **inputBIR** в пару членов Си **Form/InputBIR** выполняется в соответствии с Таблицей 30 с выполнением следующих действий (в заданном порядке):

a) установка члена **Form,** который основывается на имеющейся альтернативе компонента **inputBIR**;

b) выбор соответствующей альтернативы члена **InputBIR**;

c) преобразование компонента АСН.1 в член Си.

Таблица 30 – Преобразование данных между альтернативами члена **BIR** типа Си **BioAPI_INPUT_BIR** и альтернативами компонента АСН.1 **bir** типа **BioAPI-INPUT-BIR**

Значение члена Form	Альтернатива члена InputBIR	Альтернатива члена inputBIR	Ссылки
1(BioAPI_DATABASE_ID_INPUT)	BIRinDb	birInDB	Раздел 19 и 15.27
2 (BioAPI_BIR_HANDLE_INPUT)	BIRinBSP	birInBSP	Раздел 19 и 15.12
2 (BioAPI_BIR_HANDLE_INPUT)	BIR	bir	Раздел 19 и 15.12
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33		

15.47 Тип BioAPI_INPUT_BIR_FORM

15.47.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_INPUT_BIR_FORM;
```

15.47.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#define BioAPI_DATABASE_ID_INPUT      (1)
#define BioAPI_BIR_HANDLE_INPUT       (2)
#define BioAPI_FULLBIR_INPUT          (3)
```

15.47.3 В ПМО БиоАПИ отсутствует соответствующий тип АСН.1.

Примечание – Данный тип Си используется в **BioAPI_INPUT_BIR** (см.15.46) и определяет выбор одного из трех способов определения входящих ЗБИ во многих операциях. В соответствующем типе АСН.1 **BioAPI-INPUT-BIR**, выбор одного из трех способов определения входящих ЗБИ представлен конструкцией **CHOICE**.

15.48 Тип BioAPI_OPERATIONS_MASK

15.48.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_OPERATIONS_MASK;
```

15.48.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#define BioAPI_ENABLEEVENTS                 (0x00000001)
#define BioAPI_SUBSCRIBETOGUIEVENTS         (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)
```

15.48.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-OPERATIONS-MASK ::= BIT STRING {
        enableEvents                (0),
        subscribeToGUIEvents        (1),
        capture                     (2),
        createTemplate              (3),
        process                     (4),
        processWithAuxBir           (5),
        verifyMatch                 (6),
        identifyMatch               (7),
        enroll                      (8),
        verify                      (9),
        identify                    (10),
        import                      (11),
        presetIdentifyPopulation    (12),
        databaseOperations          (13),
        setPowerMode                (14),
        setIndicatorStatus          (15),
        getIndicatorStatus          (16),
        calibrateSensor             (17),
        utilities                   (18),
        queryUnits                  (20),
        queryBFPs                   (21),
        controlUnit                 (22)
    } (SIZE(32))
```

15.48.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют следующим образом: каждый бит битовой строки АСН.1 должен быть преобразован в бит целого значения Си без знака.

Наибольший бит битовой строки (бит 0) должен быть преобразован в наименее значимый бит целого числа без знака, соответствующий значению 0x00000001, а оставшиеся тридцать один бит должны быть преобразованы по порядку.

15.49 Тип BioAPI_OPTIONS_MASK

15.49.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_OPTIONS_MASK;
```

15.49.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#define BioAPI_RAW                          (0x00000001)
#define BioAPI_QUALITY_RAW                  (0x00000002)
#define BioAPI_QUALITY_INTERMEDIATE         (0x00000004)
#define BioAPI_QUALITY_PROCESSED            (0x00000008)
#define BioAPI_APP_GUI                      (0x00000010)
#define BioAPI_GUI_PROGRESS_EVENTS          (0x00000020)
#define BioAPI_SOURCEPRESENT                (0x00000040)
#define BioAPI_PAYLOAD                      (0x00000080)
#define BioAPI_BIR_SIGN                     (0x00000100)
#define BioAPI_BIR_ENCRYPT                  (0x00000200)
#define BioAPI_TEMPLATEUPDATE               (0x00000400)
#define BioAPI_ADAPTATION                   (0x00000800)
#define BioAPI_BINNING                      (0x00001000)
#define BioAPI_SELFCONTAINEDDEVICE          (0x00002000)
#define BioAPI_MOC                          (0x00004000)
#define BioAPI_SUBTYPE_TO_CAPTURE           (0x00008000)
#define BioAPI_SENSORBFP                    (0x00010000)
#define BioAPI_ARCHIVEBFP                   (0x00020000)
#define BioAPI_COMPARISONBFP                (0x00040000)
#define BioAPI_PROCESSINGBFP                (0x00080000)
#define BioAPI_COARSESCORES                 (0x00100000)
```

15.49.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-OPTIONS-MASK ::= BIT STRING {
                raw                 (0),
                qualityRaw          (1),
                qualityIntermediate (2),
                qualityProcessed    (3),
```

```
        appGui                  (4),
        guiProgressEvents       (5),
        sourcePresent           (6),
        payload                 (7),
        birSign                 (8),
        birEncrypt              (9),
        templateUpdate          (10),
        adaptation              (11),
        binning                 (12),
        selfContainedDevice     (13),
        moc                     (14),
        subtypeToCapture        (15),
        sensorBFP               (16),
        archiveBFP              (17),
        comparisonBFP           (18),
        processingBFP           (19),
        coarseScores            (20)
} (SIZE(32))
```

15.49.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют следующим образом: каждый бит битовой строки АСН.1 должен быть преобразован в бит целого числа Си без знака. Наибольший бит битовой строки (бит 0) должен быть преобразован в наименее значимый бит целого числа без знака (ответствующий значению 0x00000001), а оставшиеся тридцать один бит должны быть преобразованы по порядку.

15.50 Тип BioAPI_POWER_MODE

15.50.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_POWER_MODE;
```

15.50.2 Для поддерживания данного типа Си в БиоАПИ определены следующие символические константы:

```
#define BioAPI_POWER_NORMAL     (1)
#define BioAPI_POWER_DETECT     (2)
#define BioAPI_POWER_SLEEP      (3)
```

15.50.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-POWER-MODE ::= ENUMERATED {
        normal,
        detect,
        sleep,
        ...
}
```

15.50.4 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют в соответствии с таблицей 31.

Таблица 31 – Преобразование данных между индивидуальными значениями Си типа **BioAPI_POWER_MODE** и абстрактными значениями АСН.1 типа **BioAPI-POWER-MODE**

Значение типа Си	Абстрактное значение типа АСН.1
1 (BioAPI_POWER_NORMAL)	normal
2 (BioAPI_POWER_DETECT)	detect
3 (BioAPI_POWER_SLEEP)	sleep
Другие значения	Отсутствует – значение Си не преобразуется, см. раздел 33

15.51 Тип BioAPI_QUALITY

15.51.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef int8_t BioAPI_QUALITY;
```

15.51.2 В ПМО БиоАПИ соответствующий тип АСН.1 определятся следующим образом:

```
BioAPI-QUALITY ::= INTEGER (-2..100)
```

15.51.3 Преобразования между типом Си и типом АСН.1 (в обоих направлениях) выполняют согласно 14.1.2.

15.51.4 Значения качества, не входящие в диапазон от 2 до 100 – не преобразуются. В случае обнаружения таких значений выполняют действия, указанные в разделе 32.

15.52 Тип BioAPI_RETURN

15.52.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_RETURN;
```

15.52.2 Многие символические константы, определенные в ПМО БиоАПИ для поддерживания данного типа Си, определяют состояние ошибки.

15.52.3 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-RETURN ::= UnsignedInt
```

15.52.4 Преобразования между типом Си и типом АСН.1 (в обоих направлениях) выполняют согласно 15.1.5.

15.53 Тип BioAPI_STRING

15.53.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

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

15.53.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-STRING ::= UTF8String (CONSTRAINED BY
                    {--The UTF-8 encoding shall not contain any NULL characters--
                        --and shall be no longer than 268 octets--})
```

15.53.3 Преобразование типа Си в тип АСН.1 выполняют следующим образом: содержимое массива октетов, выделенного переменной Си до первого (исключительного) октета с нулевым значением, должно интерпретируют как UTF-8 кодировка строки символов. Абстрактное значение АСН.1 должно быть установлено в такую строку символов.

15.53.4 Преобразование типа АСН.1 в тип Си выполняют следующим образом: принимают L равным длине (в октетах) UTF-8 кодировки абстрактной строки АСН.1, в этом случае первые октеты $L + 1$ массива октетов, выделенного переменной Си, должны быть заполнены UTF-8 кодированием, а следующие за ними – октетом с нулевым значением.

Примечание – Ограничение, наложенное на определение типа АСН.1, гарантирует, что массив байтов Си не переполниться во время преобразования.

15.54 Тип BioAPI_TIME

15.54.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct bioapi_time {
        uint8_t Hour;
        uint8_t Minute;
        uint8_t Second;
    } BioAPI_TIME;
```

15.54.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-TIME ::= SEQUENCE {
        hour        INTEGER  (0..99),
        minute      INTEGER (0..99),
        second      INTEGER (0..99)
    }
```

15.54.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 32.

Таблица 32 – Преобразование данных между членами типа Си **BioAPI_TIME** и компонентами АСН.1 типа **BioAPI-TIME**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
Hour	hour	15.1.2
Minute	minute	15.1.2
Second	second	15.1.2

15.54.4 Абстрактное значение типа **BioAPI-TIME**, состоящего из трех компонентов **hour**, **minute** и **second** равных 99, допускается. В отличие от указанного определенного абстрактного значения, все значения **hour**, не входящие в диапазон от 0 до 23, все значения **minute**, не входящие в диапазон от 0 до 59, и все значения **second**, не входящие в диапазон от 0 до 59 не могут

быть преобразованы; при обнаружении таких значений выполняют действия, указанные в разделе 33.

15.55 Тип BioAPI_UNIT_ID

15.55.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint32_t BioAPI_UNIT_ID;
```

15.55.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-UNIT-ID ::= UnsignedInt
```

15.55.3 Преобразования между типом Си и типом АСН.1 (в обоих направлениях) выполняются согласно 15.1.5.

15.56 Тип BioAPI_UNIT_LIST_ELEMENT

15.56.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct _bioapi_unit_list_element {
        BioAPI_CATEGORY UnitCategory;
        BioAPI_UNIT_ID UnitID;
    } BioAPI_UNIT_LIST_ELEMENT;
```

15.56.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-UNIT-LIST-ELEMENT ::= SEQUENCE {
        category        BioAPI-CATEGORY,
        unitID  BioAPI-UNIT-ID
    }
```

15.56.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 33.

Таблица 33 – Преобразование данных между членами типа Си **BioAPI_UNIT_LIST_ELEMENT** и компонентами АСН.1 типа **BioAPI-UNIT-LIST-ELEMENT**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
UnitCategory	category (категория)	15.21
UnitID	unitID (ИД модуля)	15.55

15.57 Тип BioAPI_UNIT_SCHEMA

15.57.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef struct bioapi_unit_schema {
        BioAPI_UUID             BSPProductUuid;
        BioAPI_UUID             UnitManagerProductUuid;
        BioAPI_UNIT_ID          UnitId;
        BioAPI_CATEGORY         UnitCategory;
        BioAPI_UUID             UnitProperties;
        BioAPI_STRING           VendorInformation;
        BioAPI_EVENT_MASK       SupportedEvents;
        BioAPI_UUID             UnitPropertyUuid;
        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;
```

15.57.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-UNIT-SCHEMA ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID,
        unitManagerProductUuid  BioAPI-UUID,
        unitId                  BioAPI-UNIT-ID,
        category                BioAPI-CATEGORY,
```

```
    unitProperties              BioAPI-UUID,
    vendorInformation           BioAPI-STRING,
    supportedUnitEvents         BioAPI-UNIT-EVENT-TYPE-MASK,
    propertyUuid                BioAPI-UUID,
    property                    BioAPI-DATA,
    hardwareVersion             BioAPI-STRING,
    firmwareVersion             BioAPI-STRING,
    softwareVersion             BioAPI-STRING,
    hardwareSerialNumber        BioAPI-STRING,
    authenticatedHardware       BOOLEAN,
    maxBSPDbSize                UnsignedInt,
    maxIdentify                 UnsignedInt
}
```

15.57.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют путем преобразования между индивидуальными членами Си и компонентами АСН.1 в соответствии с таблицей 34.

Таблица 34 – Преобразование данных между членами типа Си **BioAPI_UNIT_SCHEMA** и компонентами АСН.1 типа **BioAPI-UNIT-SCHEMA**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
BSPProductUuid	bspProductUuid	15.58
UnitManagerProductUuid	unitManagerProductUuid	15.58
UnitId	unitId	15.55
UnitCategory	category	15.21
UnitProperties	unitProperties	15.58
VendorInformation	vendorInformation	15.53
SupportedEvents	supportedUnitEvents	15.31
UnitPropertyUuid	propertyUuid	15.58
UnitProperty	property	15.22
HardwareVersion	hardwareVersion	15.53

Окончание таблицы 34

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
FirmwareVersion	firmwareVersion	15.53
SoftwareVersion	softwareVersion	15.53
HardwareSerialNumber	hardwareSerialNumber	15.53
AuthenticatedHardware	authenticatedHardware	15.18
MaxBSPDbSize	maxBSPDbSize	15.1.5
MaxIdentify	maxIdentify	15.1.5

Примечание – В типе СИ **BioAPI_UNIT_SCHEMA** не присутствует параметр **HostingEndpointIRI** или УУИД доступа ПБУ. Все модули, возвращенные **BioAPI_QueryUnits,** находятся в той конечной точке ПМО БиоАПИ, которая является главной конечной точкой ПБУ. Схема модулей, представленная функцией **BioAPI_EVENT_HANDLER**, принадлежит модулю, который управляется (прямо или косвенно) ПБУ, чей УУИД представлен в качестве первого параметра такой функции.

15.58 Тип BioAPI_UUID

15.58.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_UUID[16];
```

15.58.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-UUID ::= OCTET STRING (SIZE(16))
```

15.58.3 Преобразование типа Си в тип АСН.1 выполняют путем установки абстрактного значения АСН.1 в октетную строку, содержащую 16 октетов в октетной строке, которая выделена переменной Си.

15.58.4 Преобразование типа АСН.1 в тип Си выполняют путем заполнения октетной строки, которая выделена переменной Си, 16 октетами строки АСН.1.

15.59 Тип BioAPI_VERSION

15.59.1 В ПМО БиоАПИ данный тип Си определен следующим образом:

```
typedef uint8_t BioAPI_VERSION;
```

15.59.2 В ПМО БиоАПИ соответствующий тип АСН.1 определен следующим образом:

```
BioAPI-VERSION ::= SEQUENCE {
        major INTEGER    (0..15),
        minor INTEGER    (0..15)
    }
```

15.59.3 Преобразование между типом Си и типом АСН.1 (в обоих направлениях) выполняют следующим образом: наиболее значимую половину октетов и наименее значимую половину октетов рассматривают как два отдельных (целых) члена типа Си, каждый из которых должен быть преобразован согласно таблице 35.

Таблица 35 – Преобразование данных между членами типа Си **BioAPI_VERSION** и компонентами АСН.1 типа **BioAPI-VERSION**

Член типа Си	Компонент типа АСН.1	Пункт настоящего стандарта
Most significant half octet	major	15.1.2
Least significant half octet	minor	15.1.2

16 Функции, определенные в БиоАПИ, и соответствующие сообщения ПМО БиоАПИ

16.1 Функция BioAPI_Init

16.1.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_Init
    (BioAPI_VERSION Version);
```

16.1.2 Связанные типы сообщений ПМО БиоАПИ отсутствуют.

16.1.3 Когда инфраструктура получает вызов к функции **BioAPI_Init** от локального приложения, она должна выполнить следующие действия в указанном порядке:

a) совершить внутренний вызов функции БиоАПИ (см. 13.10) к той же функции с теми же значениями параметра, как и во входящем вызове;

b) Если возвращенное значение внутреннего вызова не равно 0, возвратить такое значение локальному приложению без выполнения следующих действий;

c) создать все концептуальные таблицы первоначально пустыми;

d) установить ИИР конечной точки в локальной конечной точке в верное уникальное ИИР.

Примечание 1 – Настоящий стандарт не распространяется на средства, с помощью которых инфраструктура выбирает ИИР;

e) добавить поле к таблице **VisibleEndpoints** (см. 18.2), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной точки и

2) оставшиеся компоненты должны быть установлены из атрибутов схемы структуры локального реестра компонентов;

f) для каждой схемы ПБУ в локальном реестре компонентов добавить поле к таблице **VisibleBSPRegistrations** (см. 18.3), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной точки;

2) компонент **bspAccessUuid** должен быть установлен в динамически произведенный УУИД; и

3) оставшиеся компоненты должны быть установлены из атрибутов схемы ПБУ локального реестра компонентов;

Примечание 2 – Другой УУИД доступа ПБУ может быть создан от того же ПБУ, при этом каждый раз **BioAPI_Init** или **BIOAPI_InitEndpoint** вызывается локальным приложением;

g) для каждой схемы ПБФ в локальном реестре компонентов добавить поле к таблице **VisibleBFPRegistrations** (см. 18.4), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной точки; и

2) оставшиеся компоненты должны быть установлены из атрибутов схемы ПБФ локального реестра компонентов;

h) при необходимости выполнять действия, которые выполняют в случае, когда инфраструктура получает один или более вызовов функции **BioAPI_LinkToEndpoint** от локального приложения (см. 16.4), со значением параметра **slaveEndpointIRI,** основанном на данных конфигурации инфраструктуры;

i) возвратить значение 0 локальному приложению.

Примечание 3 – Настоящий стандарт не распространяется на средства, с помощью которых инфраструктура определяет набор конечных точек ПМО БиоАПИ для связи во время вызова функции **BioAPI_Init**. Эта информация может быть получена из параметров конфигурации реализации инфраструктуры.

16.2 Функция BioAPI_InitEndpoint

16.2.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_InitEndpoint
    (BioAPI_VERSION Version,
    const uint8_t *LocalEndpointIRI);
```

16.2.2 Связанные типы сообщений ПМО БиоАПИ отсутствуют.

16.2.3 Когда инфраструктура получает вызов к функции **BioAPI_InitEndpoint** от локального приложения, она должна выполнить следующие действия в указанном порядке:

a) совершить внутренний вызов функции БиоАПИ (см. 13.10) к той же функции с теми же значениями параметра, как и во входящем вызове;

b) Если возвращенное значение внутреннего вызова не равно 0, возвратить такое значение локальному приложению без выполнения следующих действий;

c) создать все концептуальные таблицы первоначально пустыми;

d) Если значение **LocalEndpointIRI** не **NULL**, установить ИИР конечной точки локальной конечной точки на такое значение;

e) Если значением **LocalEndpointIRI** является **NULL**, либо установить ИИР конечной точки локальной конечной точки на любое верное уникальное значение ИИР, либо возвратить значение **BioAPIERR_LOCAL_ENDPOINT_IRI_NEEDED** локальному приложению без выполнения указанных далее действий.

Примечание 1 – Настоящий стандарт не распространяется на средства, с помощью которых инфраструктура выбирает ИИР. Инфраструктура, которая играет второстепенную роль, должна иметь возможность установки ИИР локальной конечной точки даже при отсутствии локального приложения, которое совершает вызов вызова **BioAPI_InitEndpoint**;

f) добавить поле к таблице **VisibleEndpoints** (см. 18.2), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной конечной точки и

2) оставшиеся компоненты должны быть установлены из атрибутов схемы структуры локального реестра компонентов;

g) для каждой схемы ПБУ в локальном реестре компонентов добавить поле к таблице **VisibleBSPRegistrations** (см. 18.3), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной конечной точки;

2) компонент **bspAccessUuid** должен быть установлен в динамически созданный УУИД и

3) оставшиеся компоненты должны быть установлены из атрибутов схемы ПБУ локального реестра компонентов.

Примечание 2 – Различные УУИД доступа ПБУ могут быть сгенерированы для одного и того же ПБУ каждый раз, когда **BioAPI_Init** или **BioAPI_InitEndpoint** вызывается локальным приложением;

h) для каждой схемы ПБФ в локальном реестре компонентов добавить поле к таблице **VisibleBFPRegistrations** (см. 18.4), для которого:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной конечной точки; и

2) оставшиеся компоненты должны быть установлены из атрибутов схемы ПБФ локального реестра компонентов;

i) при необходимости выполнять действия, которые выполняют в случае, когда инфраструктура получает один или более вызовов функции **BioAPI_LinkToEndpoint** от локального приложения (см. 16.4) со значением параметра **slaveEndpointIRI**, основанном на данных конфигурации инфраструктуры;

j) возвратить значение 0 локальному приложению.

Примечание 3 – Настоящий стандарт не распространяется на средства, с помощью которых инфраструктура определяет набор конечных точек ПМО БиоАПИ для связи во время вызова функции **BioAPI_InitEndpoint**. Эта информация может быть получена из параметров конфигурации реализации инфраструктуры.

16.3 Функция BioAPI_Terminate

16.3.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_Terminate
    (void);
```

16.3.2 С данной функцией связан только тип сообщения уведомления ПМО БиоАПИ **masterDeletionEvent**. Этот тип сообщений ПМО БиоАПИ переносит значение следующего параметра типа АСН.1 сообщений ПМО БиоАПИ: **MasterDeletionEvent-NotificationParams ::= NULL**

16.3.3 Когда инфраструктура получает вызов к функции **BioAPI_Terminate** от локального приложения, она выполняет действия, указанные далее в указанном порядке.

16.3.3.1 Для каждого поля таблицы **VisibleEndpoints** (см. 18.2), в котором компонент **hostingEndpointIRI** (*slaveEndpointIRI*) имеет значение, отличающееся от ИИР локальной конечной точки, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать и отправить сообщение запроса ПМО БиоАПИ **deleteMaster** (см. 13.1 и 16.5.2) с ИИР второстепенной конечной точки, установленным в *slaveEndpointIRI*, со значением параметра, установленным в **NULL**;

b) принять корреспондирующее сообщение ответа ПМО БиоАПИ **deleteMaster** (см. 13.6) и

c) удалить поле таблицы схемы структуры (выполняют действия, указанные в 18.2.3);

Примечание – Такими второстепенными конечными точками являются все конечные точки ПМО БиоАПИ, в которые было отправлено сообщение запроса ПМО БиоАПИ **addMaster**, но не отправлено последующее сообщение запроса ПМО БиоАПИ **deleteMaster**, которое может быть любым сообщением запроса ПМО БиоАПИ **addMaster,** отправленным во время обработки вызова функции **BioAPI_Init** или **BioAPI_InitEndpoint** (см. 16.1);

d) при необходимости разрушить соединение(я) транспортного уровня с второстепенной конечной точкой согласно требованиям использующейся привязки транспортного протокола.

16.3.3.2 Для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1), инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать и отправить сообщение уведомления ПМО БиоАПИ **masterDeletionEvent** (см. 13.4) с ИИР главной конечной точки установленным из компонента **masterEndpointIRI** в *slaveEndpointIRI*, и со значением параметра, установленным в **NULL**;

b) удалить поле таблицы **MasterEndpoints** (выполняют действия, указанные в 18.1.3);

c) при необходимости, разрушить соединение(я) транспортного уровня с главной конечной точкой согласно требованиям использующейся привязки транспортного протокола.

16.3.3.3 Удалить все поля таблицы **ApplicationOwnedMemoryBlocks** (см. 18.13).

16.3.3.4 Совершить внутренний вызов функции БиоАПИ (см. 13.10) к той же функции, как при входящем вызове, и вернуть локальному приложению возвращенное значение внутреннего вызова.

16.3.3.5 Удалить все временные абстрактные значения, которые до сих пор не были удалены.

16.3.4 Когда инфраструктура принимает (см. 12.9) сообщение уведомления ПМО БиоАПИ **masterDeletionEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) удалить поля (если есть) таблицы **VisibleEndpoints,** в которых компонент **hostingEndpointIRI** содержит ИИР второстепенной конечной точки (выполняют действия, указанные в17.2.3);

b) установить, что для отправления отсутствуют сообщения подтверждение ПМО БиоАПИ.

16.4 Функция BioAPI_LinkToEndpoint

16.4.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI_LinkToEndpoint
        (const uint8_t *SlaveEndpointIRI);
```

16.4.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **addMaster** и тип сообщения ответа

ПМО БиоАПИ **addMaster**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
AddMaster-RequestParams ::= SEQUENCE {
        bipVersion          BioAPI-VERSION
}
```

и

```
AddMaster-ResponseParams ::= SEQUENCE {
        fwSchema            BioAPI-FRAMEWORK-SCHEMA OPTIONAL,
        bspSchemas          SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                bspSchema BioAPI-BSP-SCHEMA,
        bfpSchemas          SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                bfpSchema BioAPI-BFP-SCHEMA
}
```

16.4.3 Следующий тип АСН.1 помогает при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
LinkCallParams ::= SEQUENCE {
        slaveEndpointIRI EndpointIRI
}
```

16.4.4 Когда инфраструктура получает вызов к функции **BioAPI_LinkToEndpoint** от локального приложения, она должна выполнить следующие действия в указанном порядке:

a) установить число связей, связанных со связью ПМО БиоАПИ, на случайно созданное значение и идентификатор запроса, связанного со связью ПМО БиоАПИ, на другое случайно созданное значение.

Примечание – Число связей не является идентификатором связи, так как существует незначительная вероятность дублирования. Главным назначением числа связи является увеличение надежности реализаций и упрощение диагностики;

b) создать временное абстрактное значение (*linkCallParams*) типа **LinkCallParams** (см. 16.4.3) из преобразований параметров вызова функции **BioAPI_LinkToEndpoint**, согласно 16.4.6;

c) проинспектировать таблицу **VisibleEndpoints** (см. 18.2) на наличие поля, в котором компонент **hostingEndpointIRI** имеет такое же значение, как и компонент **slaveEndpointIRI** *linkCallParams*;

d) при необходимости установить одно- или двухуровневые транспортные соединения с второстепенной конечной точкой, согласно требованиям использующейся привязки транспортного протокола;

e) Если такое поле обнаружено, возвратить значение **BioAPIERR_SLAVE_ALREADY_LINKED** локальному приложению, без выполнения следующих действий;

f) создать временное абстрактное значение (*outgoingRequestParams*) типа **AddMaster-RequestParams** (см. 16.4.2), в котором компонент **bipVersion** должен быть установлен на номер версии настоящего стандарта (см. 13.15);

g) создать и отправить сообщение запроса ПМО БиоАПИ **addMaster** (см. 13.2) с ИИР второстепенной конечной точки, установленной из компонента **slaveEndpointIRI** *linkCallParams*, и значением параметра, установленным в *outgoingRequestParams*;

h) принять корреспондирующее сообщение ответа ПМО БиоАПИ **addMaster** (см. 13.6);

i) Если возвращенное значение **addMaster** сообщения ответа ПМО БиоАПИ не равно 0, возвратить такое значение локальному приложению без выполнения следующих действий;

j) разрешить *incomingResponseParams* выступать в качестве значения параметра **AddMaster-ResponseParams** (см 16.4.2) в сообщении ответа ПМО БиоАПИ **addMaster**;

k) Если добавочный компонент **fwSchema** отсутствует, возвратить значение **BioAPIERR_FRAMEWORK_SCHEMA_ABSENT** локальному приложению, без выполнения следующич действий;

l) Если в качестве значение компонента **bspProductUuid** в двух или более элементах компонента **bspSchemas** *incomingResponseParams* встречаются одинаковые УУИД продукта ПБУ, возвратить значение **BioAPIERR_DUPLICATE_BSP_PRODUCT_UUID** локальному приложению без выполнения следующич действий;

m) Если в качестве значение компонента **bfpProductUuid** в двух или более элементах компонента **bfpSchemas** *incomingResponseParams* встречаются одинаковые УУИД продукта ПБФ, возвратить значение **BioAPIERR_DUPLICATE_BFP_PRODUCT_UUID** локальному приложению без выполнения последующих действий;

n) добавить поле в таблицу **VisibleEndpoints** (см. 18.2), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен из компонента **slaveEndpointIRI** *linkCallParams* и

2) оставшиеся компоненты должны быть установлены из компонентов компонента **fwSchema** *incomingResponseParams*;

o) в каждый элемент (*bspSchema*) компонента **bspSchemas** *incomingResponseParams* добавить поле в таблицу **VisibleBSPRegistrations** (см. 18.3), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен из компонента **slaveEndpointIRI** *linkCallParams*;

2) компонент **bspAccessUuid** shall должен быть установлен на динамически созданный УУИД; и

3) оставшиеся компоненты должны быть установлены из компонентов *bspSchema*;

p) для каждого элемента (читай *bfpSchema*) компонента **bfpSchemas** *incomingResponseParams* добавить поле в таблицу **VisibleBFPRegistrations** (см. 18.4), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен из компонента **slaveEndpointIRI** *linkCallParams*; и

2) оставшиеся компоненты должны быть установлены из компонентов *bfpSchema*;

q) возвратить значение 0 локальному приложению.

16.4.5 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **addMaster** от конечной точки ПМО БиоАПИ, она должна выполнить следующие действия в указанном порядке:

a) установить число связей, связанных со связью ПМО БиоАПИ, на число связи, обнаруженное во входящем сообщении запроса ПМО БиоАПИ, и установить идентификатор запроса, связанный со связью ПМО БиоАПИ, на другое случайно созданное значение;

b) при необходимости создать и отправить корреспондирующее сообщение ответа ПМО БиоАПИ **addMaster** (см. 13.3) с возвращенным значением, установленным на **BioAPIERR_UNWILLING_TO_ACT_AS_SLAVE**, без выполнения последующих действий;

Примечание – Для посылающей запрос конечной точки это служит показателем того, что данная конечная точка отказывается от запроса связи с конечной точкой. Настоящий стандарт не устанавливает критерии принятия или отказа от запроса связи с конечной точкой;

c) Если компонент **bipVersion** значения параметра сообщения запроса ПМО БиоАПИ **addMaster** имеет значение, отличающееся от номера версии настоящего стандарта (см. 13.15), инфраструктура должна:

1) создать и отправить корреспондирующее сообщение запроса **addMaster** (см.13.3) с возвращенным значением, установленным на **BioAPIERR_INCOMPATIBLE_VERSION**, без выполнения следующих действий, либо

2) выполнить любые действия, соответствующие для такого номера версии (но не определенные в настоящем стандарте), без выполнения следующих действий;

d) проинспектировать таблицу **MasterEndpoints** (см. 18.1) на наличие поля, в котором компонент **masterEndpointIRI** содержит ИИР конечной точки запрашивающей конечной точки ПМО БиоАПИ;

e) Если такое поле обнаружено, создать и отправить корреспондирующее сообщение ответа ПМО БиоАПИ **addMaster** (см. 13.3) с возвращенным значением, установленным на **BioAPIERR_MASTER_ALREADY_LINKED** без выполнения следующих действий;

f) добавить поле в таблицу **MasterEndpoints** (см. 18.1), в котором компонент **masterEndpointIRI** должен быть установлен на ИИР конечной точки запрашивающей конечной точки;

g) определить местоположение поля (*fwSchema*) таблицы **VisibleEndpoints** (см. 18.2), в котором компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки;

h) создать временное абстрактное значение (*outgoingResponseParams*) типа **AddMaster-ResponseParams** (см. 16.4.2), в котором должен присутствовать компонент **fwSchema** установленный на *fwSchema*, а компоненты **bspSchemas** или **bfpSchemas** должны быть изначально пустыми;

i) в каждое поле (*bspSchema*) таблицы **VisibleBSPRegistrations** (см. 18.3), в котором компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки, добавить элемент в компонент **bspSchemas** *outgoingResponseParams*, в котором:

1) компонент **bspAccessUuid** должен быть установлен на 16 нулевых октетов и

2) оставшиеся компоненты элемента должны быть установлены из компонентов *bspSchema* с теми же именами;

j) в каждое поле (*bfpSchema*) таблицы **VisibleBFPRegistrations** (см.18.4), в котором компонент **hostingEndpointIRI** содержит ИИР

локальной конечной точки, добавить элемент в компонент **bfpSchemas** *outgoingResponseParams*, в котором все компоненты элемента должны быть установлены из компонентов *bfpSchema* с теми же именами;

k) создать и отправить корреспондирующее сообщение ответа ПМО БиоАПИ **addMaster** (см. 13.3) со значением параметра, установленном на *outgoingResponseParams*, и возвращенное значение, установленное на 0.

16.4.6 Преобразование параметров функции Си **BioAPI_LinkToEndpoint** в тип АСН.1 **LinkCallParams** (см. 16.4.3) выполняют путем преобразований из индивидуальных параметров функции в компоненты АСН.1 согласно таблице 36.

Таблица 36 – Преобразования данных из параметров функции **BioAPI_LinkToEndpoint** в тип АСН.1 **LinkCallParams**

Параметр функции	Компонент типа АСН.1	Подраздел настоящего стандарта
SlaveEndpointIRI	slaveEndpointIRI	15.3

16.5 Функция BioAPI_UnlinkFromEndpoint

16.5.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI_UnlinkFromEndpoint
        (const uint8_t *SlaveEndpointIRI);
```

16.5.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **deleteMaster** и тип сообщения ответа ПМО БиоАПИ **deleteMaster**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ, соответственно:

```
DeleteMaster-RequestParams ::= NULL
```

и

```
DeleteMaster-ResponseParams ::= NULL
```

16.5.3 Следующий тип АСН.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
UnlinkCallParams ::= SEQUENCE {
        slaveEndpointIRI    EndpointIRI
    }
```

16.5.4 Когда инфраструктура получает вызов к функции **BioAPI_UnlinkFromEndpoint** от локального приложения, она должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*unlinkCallParams*) типа **UnlinkCallParams** (см. 16.5.3) путем преобразований из параметров функции вызова **BioAPI_UnlinkFromEndpoint**, согласно 16.5.6;

b) разрешить *slaveEndpointIRI* иметь значение компонента **slaveEndpointIRI** *unlinkCallParams*;

c) проинспектировать таблицу **VisibleEndpoints** (см. 18.2) на наличие поля, в котором компонент **hostingEndpointIRI** имеет значение *slaveEndpointIRI*;

d) Если такого поля нет, возвратить значение **BioAPIERR_NO_SUCH_SLAVE_FOUND** локальному приложению, без выполнения следующих действий;

e) создать и отправить сообщение запроса ПМО БиоАПИ **deleteMaster** (см. 13.2) с ИИР второстепенной конечной точки, установленной на *slaveEndpointIRI*, с параметром, установленным на **NULL**;

f) принять корреспондирующее сообщение ответа ПМО БиоАПИ **deleteMaster** (см. 13.6);

g) удалить поле таблицы **VisibleEndpoints** (выполняют действия, указанные в 18.2.3);

h) Если возвращенное значение сообщения ответа ПМО БиоАПИ **deleteMaster** не равно 0, возвратить такое значение локальному приложению, без выполнения следующих действий;

i) при необходимости разрушить соединение(я) транспортного уровня с второстепенной конечной точкой согласно требованиям использующейся привязки транспортного протокола;

j) возвратить 0 локальному приложению.

16.5.5 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **deleteMaster** от главной конечной точки она должна выполнить следующие действия в указанном порядке:

a) проинспектировать таблицу **MasterEndpoints** (см. 18.1) на наличие поля, в котором компонент **masterEndpointIRI** содержит значение ИИР конечной точки запрашивающей конечной точки ПМО БиоАПИ;

b) Если такое поле не обнаружено, создать и отправить корреспондирующее сообщение ответа ПМО БиоАПИ **deleteMaster** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_NO_SUCH_MASTER_FOUND**, без выполнения следующих действий;

c) удалить поле таблицы **MasterEndpoints** (выполняют действия, указанные в 18.1.3);

d) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **deleteMaster** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращаемым значением, установленным на 0;

e) при необходимости, разрушить соединение(я) транспортного уровня с второстепенной конечной точкой согласно требованиям использующейся привязки транспортного протокола.

16.5.6 Преобразование параметров функции Си **BioAPI_UnlinkFromEndpoint** в тип АСН.1 **UnlinkCallParams** (см. 16.5.3)

выполняют путем преобразований из индивидуальных параметров функции в компоненты АСН.1 согласно таблице 37.

Таблица 37 – Преобразования данных из параметров функции **BioAPI_UnlinkFromEndpoint** в тип АСН.1 **UnlinkCallParams**

Параметр функции	Компонент типа АСН.1	Подраздел настоящего стандарта
SlaveEndpointIRI	slaveEndpointIRI	15.3

16.6 Функция BioAPI_EnumFrameworks

16.6.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI_EnumFrameworks
        (BioAPI_FRAMEWORK_SCHEMA **FwSchemaArray,
        unit32_t *NumberOfElements);
```

16.6.2 Связанные типы сообщений ПМО БиоАПИ отсутствуют.

16.6.3 Следующий тип АСН.1 применяется при для спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
EnumFrameworksCallOutputParams ::= SEQUENCE OF BioAPI-FRAMEWORK-SCHEMA
```

16.6.4 Когда инфраструктура получает вызов к функции **BioAPI_EnumFrameworks** от локального приложения, она должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*outgoingResponseParams*) типа **EnumFrameworksCallOutputParams** (см. 16.6.3) изначально пустое;

b) добавить каждое поле таблицы **VisibleEndpoints** (см. 18.2) к *outgoingResponseParams*;

c) установить исходящие параметры вызова функции **BioAPI_EnumFrameworks** путем преобразования из *outgoingResponseParams*, согласно 16.6.5;

d) возвратить значение 0 локальному приложению.

16.6.5 Преобразование параметров функции Си **BioAPI_EnumFrameworks** в тип ACH.1 **EnumFrameworksCallOutputParams** (см. 16.6.3) выполняют путем преобразования индивидуальных параметров функции в компоненты ACH.1 согласно таблице 38.

Таблица 38 – Преобразования данных из типа ACH.1 **EnumFrameworksCallOutputParams** в параметры функции **BioAPI_EnumFrameworks**

Компонент типа ACH.1	Параметр функции	Раздел, пункт настоящего стандарта
fwSchemas	FwSchemaArray, NumberOfElements	Раздел 20 и 16.6.6

16.6.6 Преобразование компонента ACH.1 **fwSchemas** в пару переменных Си, выделенных параметрами **FwSchemaArray/NumberOfElements**, выполняют следующим образом: принимают *N* равным числу элементов компонента **fwSchemas**; в этом случае новообразованный массив *N* элементов типа **BioAPI_FRAMEWORK_SCHEMA** (см. 15.33) должен быть заполнен путем преобразования каждого элемента компонента **fwSchemas**, по порядку, в элемент массива согласно 15.33. Переменная Си, выделенная параметром **FwSchemaArray** устанавливается в адрес массива, а переменная Си, выделенная параметром **NumberOfElements**, устанавливается в *N*.

16.7 Функция BioAPI_EnumBSPs

16.7.1 Данная функция определена в БиоАПИ следующим образом:

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

16.7.2 Связанные типы сообщений ПМО БиоАПИ отсутствуют.

16.7.3 Следующий тип АСН.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
EnumBSPsCallOutputParams ::= SEQUENCE OF BioAPI-BSP-SCHEMA
```

16.7.4 Когда инфраструктура получает вызов к функции **BioAPI_EnumBSPs** от локального приложения, она должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*outgoingResponseParams*) типа **EnumBSPsCallOutputParams** (см. 16.7.3) изначально пустое;

b) добавить каждое поле таблицы **VisibleBSPRegistrations** (см. 18.3) к *outgoingResponseParams*;

c) установить исходящие параметры вызова функции **BioAPI_EnumBSPs** путем преобразования *outgoingResponseParams* согласно 16.7.5;

d) возвратить значение 0 локальному приложению.

16.7.5 Преобразования параметров функции Си **BioAPI_EnumBSPs** в тип АСН.1 **EnumBSPsCallOutputParams** (см. 16.7.3) выполняют путем преобразования индивидуальных параметров функции в компоненты АСН.1 согласно таблице 39.

Таблица 39 – Преобразования данных из типа ACH.1 **EnumBSPsCallOutputParams** в параметры функции **BioAPI_EnumBSPs**

Компонент типа ACH.1	Параметр функции	Раздел, пункт настоящего стандарта
bspSchemas	BSPSchemaArray, NumberOfElements	Раздел 20 совместно с 16.7.6

16.7.6 Преобразование компонента ACH.1 **bspSchemas** в пару переменных Си, выделенных параметрами **BSPSchemaArray/NumberOfElements**, выполняют следующим образом: принимают *N*, равным числу элементов компонента **bspSchemas**; в этом случае новый массив *N* элементов типа **BioAPI_BSP_SCHEMA** (см. 15.19) должен быть заполнен путем преобразования каждого элемента компонента **bspSchemas** по порядку в элемент массива согласно 15.19. Переменная Си, выделенная параметром **BSPSchemaArray** устанавливается в адрес массива, а переменная Си, выделенная параметром **NumberOfElements**, устанавливается в *N*.

16.8 Функция BioAPI_EnumBFPs

16.8.1 Данная функция определена в БиоАПИ следующим образом:

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

16.8.2 Связанные типы сообщений ПМО БиоАПИ отсутствуют.

16.8.3 Следующий тип ACH.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
EnumBFPsCallOutputParams ::= SEQUENCE OF BioAPI-BFP-SCHEMA
```

16.8.4 Когда инфраструктура получает вызов к функции **BioAPI_EnumBFPs** от локального приложения, она должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (читай *outgoingResponseParams*) типа **EnumBFPsCallOutputParams** (см. 16.8.3) изначально пустое;

b) добавить каждое поле таблицы **VisibleBFPRegistrations** (см. 18.4) к *outgoingResponseParams*;

c) установить исходящие параметры вызова функции **BioAPI_EnumBFPs** путем преобразования *outgoingResponseParams* согласно 16.8.5;

d) возвратить значение 0 локальному приложению.

16.8.5 Преобразование параметров функции Си **BioAPI_EnumBFPs** в тип АСН.1 **EnumBFPsCallOutputParams** (см. 16.8.3) выполняют путем преобразования индивидуальных параметров функции в компоненты АСН.1 согласно таблице 40.

Таблица 40 – Преобразования данных из типа АСН.1 **EnumBFPsCallOutputParams** в параметры функции **BioAPI_EnumBFPs**

Компонент типа АСН.1	Параметр функции	Раздел, пункт настоящего стандарта
bfpSchemas	BFPSchemaArray, NumberOfElements	Раздел 20 совместно с 16.8.6

16.8.6 Преобразование компонента АСН.1 **bfpSchemas** в пару переменных Си, выделенных параметрами **BFPSchemaArray/NumberOfElements**, выполняют следующим образом: принимают *N* равным числу элементов компонента **bfpSchemas**; в этом случае новый массив *N* элементов типа **BioAPI_BFP_SCHEMA** (см. 15.5)

должен быть заполнен путем преобразования каждого элемента компонента **bfpSchemas**, по порядку, в элемент массива согласно 15.5. Переменная Си, выделенная параметром **BFPSchemaArray**, устанавливается в адрес массива, а переменная Си, выделенная параметром **NumberOfElements**, устанавливается в *N*.

16.9 Функция BioAPI_BSPLoad

16.9.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_BSPLoad
        (const BioAPI_UUID *BSPUuid,
        BioAPI_EVENT_HANDLER EventHandler,
        void* EventHandlerCtx);
```

16.9.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **bspLoad** и тип сообщения ответа **bspLoad**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
BSPLoad-RequestParams ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID,
        unitEventSubscription           BOOLEAN
    }
```

и

```
BSPLoad-ResponseParams ::= NULL
```

16.9.3 Следующий тип АСН.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
BSPLoadCallParams ::= SEQUENCE {
        bspUuid                 BioAPI-UUID,
        unitEventHandlerAddress MemoryAddress,
        unitEventHandlerContext MemoryAddress
    }
```

16.9.4 Когда инфраструктура получает запрос к функции **BioAPI_BSPLoad** от локального приложения, она сначала должна определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра

BSPUuid согласно разделу 23, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.9.4.1 Если главная конечная точка является локальной конечной точкой, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*bspLoadCallParams*) типа **BSPLoadCallParams** (см. 16.9.3) путем преобразования параметров вызова функции **BioAPI_BSPLoad** согласно 16.9;

b) добавить поле в таблицу **RunningBSPLocalReferences** (см. 18.5), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен на ИИР локальной конечной точки;

2) компонент **bspProductUuid** должен быть установлен на *bspProductUuid*;

3) Если компонент **bspUuid** *bspLoadCallParams* имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля должен быть установлен на **TRUE**; в противном случае его устанавливают на **FALSE** и

4) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** должны быть установлены из компонентов *bspLoadCallParams* с такими же именами.

Примечание – Перенос совершения внутреннего вызова функции на этап, следующий после добавления поля в таблицу, позволяет регистрировать локальному приложению любые операции модуля, происходящие во время внутреннего вызова функции;

c) совершить внутренний вызов функции БиоАПИ (см. 13.10) к той же функции с тем же значениями параметра, как и во входящем вызове, за исключением параметра **BSPUuid**, который должен быть установлен путем преобразования из *bspProductUuid* согласно разделу 19 и 15.58;

d) Если значение параметра внутреннего вызова не равно 0, удалить поле, добавленное над таблицей **RunningBSPLocalReferences**, и возвратить такое значение локальному приложению без выполнения следующих действий;

e) возвратить значение 0 локальному приложению.

16.9.4.2 Если главная конечная точка является второстепенной конечной точкой инфраструктуры, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (читай *bspLoadCallParams*) типа **BSPLoadCallParams** (см. 16.9.3) путем преобразования параметров вызова функции **BioAPI_BSPLoad** согласно 16.9;

b) добавить поле в таблицу **RunningBSPLocalReferences** (см. 18.5), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен на ИИР главенствующей конечной точки;

2) компонент **bspProductUuid** должен быть установлен на *bspProductUuid*;

3) Если компонент **bspUuid** *bspLoadCallParams* имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля должен быть установлен на **TRUE**; в противном случае, он устанавливается на **FALSE** и

4) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** должны быть установлены из компонентов *bspLoadCallParams* с такими же именами.

Примечание – Перенос запроса на этап, следующий после добавления поля в таблицу, позволяет локальному приложению регистрировать любые операции модуля, зарегистрированные второстепенной конечной точкой во время обработки запроса;

c) создать временное абстрактное значение (*outgoingRequestParams*) типа **BSPLoad-RequestParams** (см. 16.9.2), в котором:

1) компонент **bspProductUuid** должен быть установлен на *bspProductUuid* и

2) Если компонент **unitEventHandlerAddress** *bspLoadCallParams* имеет значение, отличающееся от 0, компонент **unitEventSubscription** *outgoingRequestParams* должен быть установлен на **TRUE**; в противном случае, его устанавливают на **FALSE**;

d) создать и отправить сообщение запроса ПМО БиоАПИ **bspLoad** (см. 13.2) с ИИР второстепенной локальной точки, установленном на ИИР главной конечной точки, и со значением параметра, установленном на *outgoingRequestParams*;

e) принять соответствующее сообщение ответа ПМО БиоАПИ **bspLoad** (см. 13.6);

f) Если возвращенное значение сообщении ответа ПМО БиоАПИ не равно 0, удалить поле, добавленное над таблицей **RunningBSPLocalReferences**, и возвратить такое значение локальному приложению без выполнения следующих действий;

g) возвратить 0 локальному приложению.

16.9.4.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.9.5 Когда инфраструктура получает (см. 13.9) сообщение запроса ПМО БиоАПИ **bspLoad** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* иметь значение параметра типа **BSPLoad-RequestParams** (см. 16.9.2) в сообщении запроса ПМО БиоАПИ **bspLoad**;

b) добавить поле в таблицу **RunningBSPRemoteReferences** (см. 18.6), в котором компонент **referrerEndpointIRI** должен быть установлен на ИИР главной конечной точки, а компоненты **bspProductUuid** и **unitEventSubscription** должны быть установлены из компонентов *incomingRequestParams* с такими же именами.

Примечание – Перенос внутреннего вызова функции на этап, следующий после добавления поля в таблицу, позволяет регистрировать главной конечной точке любые операции модуля, происходящие во время внутреннего вызова функции;

c) создать временное абстрактное значение (*bspLoadCallParams*) типа **BSPLoadCallParams** (см. 16.9.3), в котором:

1) компонент **bspUuid** должны быть установлены из компонента **bspProductUuid** *incomingRequestParams* и

2) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** должны быть установлены на 0;

d) совершить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPLoad**, в котором параметры вызова функции должны быть установлены путем преобразования *bspLoadCallParams* согласно 16.9;

e) Если возвращенное значение внутреннего вызова не равно 0, удалить поле, добавленное над таблицей **RunningBSPRemoteReferences**, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspLoad** (см. 13.3) с возвращаемым значением установленным на такое значение без выполнения следующих действий;

f) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspLoad** (см. 13.3) со значением параметра, установленным на *outgoingResponseParams,*и возвращаемым значением, установленным на 0.

16.9.6 Преобразование между параметрами функции Си **BioAPI_BSPLoad** и типом АСН.1 **BSPLoadCallParams** (см. 16.9.3) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 41.

Таблица 41 – Преобразования данных между параметрами функции **BioAPI_BSPLoad** и типом АСН.1 **BSPLoadCallParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPUuid	bspUuid	Раздел 19 совместно с 15.58
EventHandler	unitEventHandlerAddress	15.1.7
EventHandlerCtx	unitEventHandlerContext	15.1.7

16.10 Функция BioAPI_BSPUnload

16.10.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_BSPUnload
        (const BioAPI_UUID *BSPUuid,
        BioAPI_EVENT_HANDLER EventHandler,
        void* EventHandlerCtx);
```

16.10.2 С данной функцией связана два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **bspUnload** и тип сообщения ответа **bspUnload**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
BSPUnload-RequestParams ::= SEQUENCE {
                bspProductUuid BioAPI-UUID,
                unitEventSubscription BOOLEAN
        }
```

и

```
BSPUnload-ResponseParams ::= NULL
```

16.10.3 Следующий тип АСН.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
BSPUnloadCallParams ::= SEQUENCE {
                bspUuid BioAPI-UUID,
```

unitEventHandlerAddress MemoryAddress,
unitEventHandlerContext MemoryAddress
}

16.10.4 Когда инфраструктура получает запрос к функции **BioAPI_BSPUnload** от локального приложения, она сначала должна определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.10.4.1 Если главная конечная точка является локальной конечной точкой, инфраструктура должна выполнить следующие действия в указанном порядке:

a) совершить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с такими же значениями параметра, как во входящем вызове, за исключением параметра **BSPUuid**, который должен быть установлен путем преобразования из *bspProductUuid* согласно разделу 19 совместно с 15.58;

b) Если возвращенное значение внутреннего вызова не равно 0, возвратить такое значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*bspUnloadCallParams*) типа **BSPUnloadCallParams** (см. 16.10.3) путем преобразования параметров вызова функции **BioAPI_BSPUnload** согласно 16.10.6;

d) проверить таблицу **RunningBSPLocalReferences** (см. 18.5) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки;

2) компонент **bspProductUuid** имеет значение *bspProductUuid*;

3) Если компонент **bspUuid** *bspUnloadCallParams* имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid**

поля имеет значение **TRUE**; в противном случае, он имеет значение **FALSE** и

4) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** имеют такие же значения, как и компоненты *bspUnloadCallParams* с такими же именами;

e) Если такое поле не обнаружено, возвратить значение **BioAPIERR_NOT_A_RUNNING_BSP** локальному приложению без выполнения следующих действий;

f) удалить поле таблицы **RunningBSPLocalReferences** (выполняют действия, указанные в 18.5.3);

Примечание – Если обнаружено несколько полей, любое из них (только одно) будет удалено.

g) возвратить значение 0 локальному приложению.

16.10.4.2 Если главная конечная точка является второстепенной конечной точкой инфраструктуры, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*outgoingRequestParams*) типа **BSPUnload-RequestParams** (см. 16.10.2), в котором:

1) компонент **bspProductUuid** должен быть установлен на *bspProductUuid* и

2) Если компонент **unitEventHandlerAddress** имеет значение, отличающееся от 0, компонент **unitEventSubscription** *outgoingRequestParams* должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

b) создать и отправить сообщение запроса ПМО БиоАПИ **bspUnload** (см. 13.2) с ИИР второстепенной конечной точки, установленным на ИИР главной конечной точки, и со значением параметра, установленном на *outgoingRequestParams*;

c) принять соответствующее сообщение ответа ПМО БиоАПИ **bspUnload** (см. 13.6);

d) Если возвращенное значение сообщения ответа ПМО БиоАПИ не равно 0, возвратить такое значение локальному приложению без выполнения следующих действий;

e) создать временное абстрактное значение (*bspUnloadCallParams*) типа **BSPUnloadCallParams** (см. 16.10.3) путем преобразования параметров вызова функции **BioAPI_BSPUnload** согласно 16.10.6;

f) проверить таблицу **RunningBSPLocalReferences** (см. 18.5) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР главной конечной точки;

2) компонент **bspProductUuid** имеет значение *bspProductUuid*;

3) Если компонент **bspUuid** *bspUnloadCallParams* имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля имеет значение **TRUE**; в противном случае он имеет значение **FALSE** и

4) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** имеют такие же значения, как и компоненты *bspUnloadCallParams* с такими же именами;

g) Если такое поле не обнаружено, отправить значение **BioAPIERR_NOT_A_RUNNING_BSP** локальному приложению без выполнения следующих действий.

h) удалить поле таблицы **RunningBSPLocalReferences** (выполняют действия, указанные в 18.5.3);

Примечание – Если обнаружено несколько полей, любое из них (только одно) будет удалено.

i) возвратить значение 0 локальному приложению.

16.10.4.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.10.5 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **bspUnload** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* иметь значение параметра типа **BSPUnload-RequestParams** (см. 16.10.2) сообщения запроса ПМО БиоАПИ **bspUnload**;

b) создать временное абстрактное значение (*bspUnloadCallParams*) типа **BSPUnloadCallParams** (см. 16.10.3), в котором:

1) компонент **bspUuid** должен быть установлен из компонента **bspProductUuid** *incomingRequestParams* и

2) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** должны быть установлены на 0;

c) совершить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPUnload**, в котором параметры вызова функции должны быть установлены путем преобразования *bspUnloadCallParams* согласно 16.10.6;

d) Если возвращенное значение внутреннего вызова не является 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspUnload** (см. 13.3) с возвращаемым значением, установленным на такое значение без выполнения следующих действий;

e) проверить таблицу **RunningBSPRemoteReferences** (см. 18.6) на наличие поля, в котором компонент **referrerEndpointIRI** содержит ИИР главной конечной точки, а компоненты **bspProductUuid** и **unitEventSubscription** имеют такие же значения, как и компоненты *incomingRequestParams* с такими же именами;

f) Если такое поле не обнаружено, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspUnload** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_NOT_A_RUNNING_BSP** без выполнения следующих действий;

g) удалить поле таблицы **RunningBSPRemoteReferences** (выполняют действия, указанные в 18.6.3).

Примечание – Если обнаружено несколько полей, любое из них (только одно) будет удалено;

h) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspUnload** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращаемым значением, установленным на 0.

15.10.6 Преобразование между параметрами функции Си **BioAPI_BSPUnload** и типом АСН.1 **BSPUnloadCallParams** (см. 16.10.3) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 42.

Таблица 42 – Преобразования данных между параметрами функции **BioAPI_BSPUnload** и типом АСН.1 **BSPUnloadCallParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPUuid	bspUuid	Раздел 19 совместно с 15.58
EventHandler	unitEventHandlerAddress	15.1.7
EventHandlerCtx	unitEventHandlerContext	15.1.7

16.11 Функция BioAPI_QueryUnits

16.11.1 Данная функция определена в БиоАПИ следующим образом:

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

16.11.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **queryUnits** и тип сообщения ответа

queryUnits, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
QueryUnits-RequestParams ::= SEQUENCE {
        bspProductUuid      BioAPI-UUID
}
```

и

```
QueryUnits-ResponseParams ::= SEQUENCE {
        unitSchemas         SEQUENCE (SIZE(0..max-unsigned-int)) OF
        unitSchema          BioAPI-UNIT-SCHEMA
}
```

16.11.3 Когда инфраструктура получает вызов к функции **BioAPI_QueryUnits** от локального приложения, она должна сначала определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23. Если главная конечная точка является локальной, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPUuid**, который должен быть установлен путем преобразования *bspProductUuid* согласно разделу 19 совместно с 15.58, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главная конечная точка является второстепенной конечной точкой инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двух сообщений запроса/ответа ПМО БиоАПИ **queryUnits** согласно разделу 27, выполняя действия, указанные в 16.11.5 и 16.11.6 для преобразования между параметрами функции и компонентами АСН.1, в том случае, если это требование указано в данном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.11.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **queryUnits** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к

BioAPI_QueryUnits для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **queryUnits** согласно разделу 28, выполняя действия, указанные в 16.11.5 и 16.11.6 для преобразования между параметрами функции и компонентами АСН.1, если это требование указано в данном разделе разделу.

16.11.5 Преобразования между параметрами функции Си **BioAPI_QueryUnits** и типом АСН.1 **QueryUnits-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 43.

Таблица 43 – Преобразования данных между параметрами функции **BioAPI_QueryUnits** и типом АСН.1 **QueryUnits-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел настоящего стандарта
BSPUuid	bspProductUuid	Раздел 25
UnitSchemaArray	*отсутствует*	Раздел 22
NumberOfElements	*отсутствует*	Раздел 22

16.11.6 Преобразование между параметрами функции Си **BioAPI_QueryUnits** и типом АСН.1 **QueryUnits-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 44.

Таблица 44 – Преобразования данных между параметрами функции **BioAPI_QueryUnits** и типом АСН.1 **QueryUnits-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
UnitSchemaArray, NumberOfElements	unitSchemas	раздел 20 совместно с 16.11.7 и 16.11.8

16.11.7 Преобразование двух переменных Си, выделенных параметрами **UnitSchemaArray/NumberOfElements**, в компонент АСН.1 **unitSchemas** выполняют следующим образом: принимаем *N* равным значению переменной Си, выделенному параметром **NumberOfElements**, в этом случае каждый первый элемент *N* (типа **BioAPI_UNIT_SCHEMA** – см. 15.57) в массиве, выделенном переменной Си, которая выделена параметром **UnitSchemaArray**, должен быть преобразован по порядку в элемент компонента **unitSchemas** согласно 15.57. Компонент **unitSchemas** должен иметь точное число *N* элементов.

16.11.8 Преобразование компонента АСН.1 **unitSchemas** в два переменных Си, выделенных параметрами **UnitSchemaArray/NumberOfElements**, выполняют следующим образом: принимают *N* равным числу элементов компонента **unitSchemas**; в этом случае новый массив *N* элементов типа **BioAPI_UNIT_SCHEMA** (см. 15.57) должен быть заполнен путем преобразования каждого элемента компонента **unitSchemas** по порядку в элемент массива согласно 15.57. Переменная Си, выделенная параметром **UnitSchemaArray**, должна быть установлена в адрес массива, а переменная Си, выделенная параметром **NumberOfElements**, должна быть установлена в *N*.

16.12 Функция BioAPI_QueryBFPs

16.12.1 Данная функция определена в БиоАПИ следующим образом:

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

16.12.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **queryBFPs** и тип сообщения ответа **queryBFPs**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
QueryBFPs-RequestParams ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID
    }
```

и

```
QueryBFPs-ResponseParams ::= SEQUENCE {
        bfps                    SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                    bfp BioAPI-BFP-LIST-ELEMENT
    }
```

16.12.3 Когда инфраструктура получает вызов к функции **BioAPI_QueryBFPs** от локального приложения, она должна сначала определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23. Если главная конечная точка является локальной конечной точкой, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как при выполнении входящего вызова, за исключением параметра **BSPUuid**, который должен быть установлен путем преобразования *bspProductUuid* согласно разделу 19 совместно с 15.58, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главная конечная точка является второстепенной конечной точкой инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **queryBFPs** согласно разделу 27, выполняя при этом действия, указанные в

16.12.5 и 16.12.6 для преобразования между параметрами функции и компонентами АСН.1, если это установленно в данном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.12.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **queryBFPs** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_QueryBFPs** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **queryBFPs** согласно разделу 28, выполняя действия, указанные в 16.12.5 и 16.12.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в данном разделе.

16.12.5 Преобразование между параметрами функции Си **BioAPI_QueryBFPs** и типом АСН.1 **QueryBFPs-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 45.

Таблица 45 – Преобразования данных между параметрами функции **BioAPI_QueryBFPs** и типом АСН.1 **QueryBFPs-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел настоящего стандарта
BSPUuid	bspProductUuid	Раздел 25
BFPList	Отсутствует	Раздел 22
NumberOfElements	Отсутствует	Раздел 22

16.12.6 Преобразование между параметрами функции Си **BioAPI_QueryBFPs** и типом АСН.1 **QueryBFPs-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 46.

Таблица 46 – Преобразования данных между параметрами функции **BioAPI_QueryBFPs** и типом АСН.1 **QueryBFPs-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BFPList, NumberOfElements	bfps	Раздел 20 совместно с 16.12.7 и 16.12.8

16.12.7 Преобразование двух переменных Си, выделенных параметрами **BFPList/NumberOfElements**, в компонент АСН.1 **bfps** выполняют следующим образом: принимают *N* равным значению переменной Си, выделенному параметром **NumberOfElements**; в этом случае каждый первый элемент *N* типа **BioAPI_BFP_LIST_ELEMENT** (см. 15.4) в массиве, выделенном переменной Си, которая выделена параметром **BFPList**, должен быть преобразован по порядку в элемент компонента **bfps** согласно 15.4. Компонент **bfps** должен иметь точное *N* элементов.

16.12.8 Преобразование компонента АСН.1 **bfps** в пару переменных Си, выделенных параметрами **BFPList/NumberOfElements**, выполняется следующим образом: Принимаем *N* за число элементов компонента **bfps**, в этом случае новый массив *N* элементов типа **BioAPI_BFP_LIST_ELEMENT** (см. 15.4) должен быть заполнен путем преобразования каждого элемента компонента **bfps** по порядку в элемент массива согласно 15.4. Переменная Си, выделенная параметром **BFPList** , должна быть установлена в адрес массива, а переменная Си, выделенная параметром **NumberOfElements**, должна быть установлена в *N*.

16.13 Функция BioAPI_BSPAttach

16.13.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_BSPAttach
        (const BioAPI_UUID *BSPUuid,
        BioAPI_VERSION Version,
        const BioAPI_UNIT_LIST_ELEMENT *UnitList,
```

```
        uint32_t NumUnits,
        BioAPI_HANDLE *NewBSPHandle);
```

16.13.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **bspAttach** и тип сообщения ответа **bspAttach**, которые переносят значение следующего параметра типа АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
BSPAttach-RequestParams ::= SEQUENCE {
        bspProductUuid      BioAPI-UUID,
        version             BioAPI-VERSION,
        units               SEQUENCE
                                (SIZE(0..max-unsigned-int)) OF
                                    unit BioAPI-UNIT-LIST-ELEMENT
    }
```

и

```
BSPAttach-ResponseParams ::= SEQUENCE {
        newOriginal         BSPHandle BioAPI-HANDLE
    }
```

16.13.3 Следующий тип АСН.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
BSPAttachCallOutputParams ::= SEQUENCE {
        newBSPHandle            BioAPI-HANDLE
    }
```

16.13.4 Когда инфраструктура получает вызов к функции **BioAPI_BSPAttach** от локального приложения, она должна сначала определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.13.4.1 Если главная конечная точка является локальной конечной точкой, инфраструктура должна выполнить следующие действия в указанном порядке:

а) совершить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с такими же значениями параметра, как и во входящем вызове,

за исключением параметра **BSPUuid**, который должен быть установлен путем преобразования из *bspProductUuid* согласно разделу 18 совместно с 15.58;

b) Если возвращенное значение внутреннего вызова не равно 0, возвратить такое значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*incomingResponseParams*) типа **BSPAttach-ResponseParams** (см. 16.13.2) путем преобразования параметров внутреннего вызова согласно 16.13.9;

d) создать временное абстрактное значение (*incomingRequestParams*) типа **BSPAttach- RequestParams** (см. 16.13.2) путем преобразования параметров вызова функции **BioAPI_BSPAttach** согласно 16.13.6;

e) создать временное абстрактное значение (*localBSPHandle*) типа **BioAPI-HANDLE** (см. 15.42), который может являться любым значением этого типа ACH.1 отличающимся от значения компонента **localBSPHandle** во всех текущих полях таблицы **AttachSessionLocalReferences** (см. 18.8).

Примечание – Настоящий стандарт не распространяется на способ генерации такого абстрактного значения;

f) добавить поле в таблицу **AttachSessionLocalReferences** (см. 18.8), в котором:

1) компонент **hostingEndpointIRI** установлен на ИИР главной конечной точки;

2) компонент **bspProductUuid** установлен на *bspProductUuid*;

3) Если компонент **bspUuid** *incomingRequestParams* имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля устанавливается на **TRUE**; в противном случае, его устанавливают на **FALSE**;

4) компонент **originalBSPHandle** устанавливают из компонента **newOriginalBSPHandle** *incomingResponseParams* и

5) компонент **localBSPHandle** устанавливается на *localBSPHandle*;

g) создать временное абстрактное значение (*outgoingResponseParams*) типа **BSPAttachCallOutputParams** (см. 16.13.3), в котором компонент **newBSPHandle** устанавливают на *localBSPHandle*;

h) установить исходящие параметры вызова функции **BioAPI_BSPAttach** путем преобразования из *outgoingResponseParams* согласно 16.13.10;

i) возвратить значение 0 локальному приложению.

16.13.4.2 Если главная конечная точка является второстепенной конечной точкой инфраструктуры, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*incomingRequestParams*) типа **BSPAttach-RequestParams** (см. 16.13.2) путем преобразования параметров вызова функции **BioAPI_BSPAttach** согласно 16.13.6;

b) создать и отправить сообщение запроса ПМО БиоАПИ **bspAttach** (см. 13.2) с ИИР второстепенной конечной точки, установленной на ИИР главной конечной точки, со значением параметра, установленным на *incomingRequestParams*;

c) принять соответствующее сообщение ответа ПМО БиоАПИ **bspAttach** (см. 13.6);

d) Если возвращенное значение сообщения ответа ПМО БиоАПИ не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

e) разрешить *incomingResponseParams* выступать в качестве значения параметра типа **BSPAttach-ResponseParams** (см. 16.13.2) сообщения ответа ПМО БиоАПИ **bspAttach**;

f) создать временное абстрактное значение (*localBSPHandle*) типа **BioAPI-HANDLE** (см. 15.42), который может быть любым значением

такого типа ACH.1, отличающимся от значения компонента **localBSPHandle** во всех текущих полях таблицы **AttachSessionLocalReferences** (см. 18.8).

Примечание – Настоящий стандарт не распространяется на способ генерации такого абстрактного значения;

g) добавить поле в таблицу **AttachSessionLocalReferences** (см. 18.8), в котором:

1) компонент **hostingEndpointIRI** установлен на ИИР главной конечной точки;

2) компонент **bspProductUuid** установлен на *bspProductUuid*;

3) Если компонент **bspUuid** *incomingRequestParams* имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля должен быть установлен на **TRUE**; в противном случае должен быть установлен на **FALSE**;

4) компонент **originalBSPHandle** устанавливают из компонента **newOriginalBSPHandle** *incomingResponseParams*; и

5) компонент **localBSPHandle** устанавливается на *localBSPHandle*;

h) создать временное абстрактное значение (*outgoingResponseParams*) типа **BSPAttachCallOutputParams** (см. 16.13.3), в котором компонент **newBSPHandle** должен быть установлен на *localBSPHandle*;

i) установить исходящие параметры вызова функции **BioAPI_BSPAttach** путем преобразования из *outgoingResponseParams* согласно 16.13.10;

j) возвратить значение 0 локальному приложению.

16.13.4.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.13.5 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **bspAttach** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **BSPAttach-RequestParams** (см. 16.13.2) сообщения запроса ПМО БиоАПИ **bspAttach;**

b) совершить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPAttach**, в котором параметры вызова функции должны быть установлены путем преобразования из *incomingRequestParams* согласно 16.13.6;

c) Если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspAttach** (см. 13.3) с возвращаемым значением, установленным на такое значение, без выполнения следующих действий;

d) создать временное абстрактное значение (*incomingResponseParams*) типа **BSPAttach-ResponseParams** (см. 16.13.2) путем преобразования параметров внутреннего вызова функции согласно 16.13.9;

e) добавить поле в таблицу **AttachSessionRemoteReferences** (см. 18.9), в котором:

1) компонент **referrerEndpointIRI** установлен на ИИР главной конечной точки;

2) компонент **bspProductUuid** установлен из компонента **bspProductUuid** *incomingRequestParams* и

3) компонент **originalBSPHandle** установлен из компонента **newOriginalBSPHandle** *incomingResponseParams*;

f) создать и отправить сообщение ответа ПМО БиоАПИ **bspAttach** (см. 13.3) со значением параметра, установленным на *incomingResponseParams*, и возвращаемым значением, установленным на 0.

16.13.6 Преобразование между параметрами функции Си **BioAPI_BSPAttach** и типом АСН.1 **BSPAttach-RequestParams** (см. 16.13.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 47.

Таблица 47 – Преобразования данных между параметрами функции **BioAPI_BSPAttach** и типом АСН.1 **BSPAttach-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPUuid	bspProductUuid	Раздел 25
Version	version	15.59
UnitList, NumUnits	units	16.13.7
NewBSPHandle	Отсутствует	Раздел 22

16.13.7 Преобразование двух параметров СИ **UnitList/NumUnits** в компонент АСН.1 **units** выполняют следующим образом: принимаем *N*, равным значению параметра **NumUnits**; в этом случае каждый первый элемент *N* типа **BioAPI_UNIT_LIST_ELEMENT** (см. 15.56) в массиве, выделенном параметром **UnitList**, должен быть преобразован по порядку в элемент компонента **units** согласно 15.56. Компонент **units** должен иметь точное число *N* элементов.

16.13.8 Преобразование компонента АСН.1 **units** в два параметра Си **UnitList/NumUnits**, выполняют следующим образом: принимают *N*, равным числу элементов компонента **units**; в этом случае новый массив *N* элементов типа **BioAPI_UNIT_LIST_ELEMENT** (см. 15.56) должен быть заполнен путем преобразования каждого элемента компонента **units** по порядку в элемент массива согласно 15.56. Параметр Си **UnitList** должен быть установлен в адрес массива, а параметр Си **NumUnits** должен быть установлен в *N*.

16.13.9 Преобразование между параметрами функции Си **BioAPI_BSPAttach** и типом АСН.1 **BSPAttach- ResponseParams** (см. 16.13.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 48.

Таблица 48 – Преобразование данных между параметрами функции **BioAPI_BSPAttach** и типом АСН.1 **BSPAttach-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
NewBSPHandle	newOriginalBSPHandle	Раздел 29 совместно с 15.42

16.13.10 Преобразование типа АСН.1 **BSPAttach-ResponseParams** (см. 16.13.2) в параметры функции Си **BioAPI_BSPAttach** выполняют путем преобразования между индивидуальными компонентами АСН.1 в параметры функции согласно таблице 49.

Таблица 49 – Преобразования данных типа АСН.1 **BSPAttach-ResponseParams** в параметры функции **BioAPI_BSPAttach**

Компонент типа АСН.1	Параметр функции	Раздел, пункт настоящего стандарта
newBSPHandle	NewBSPHandle	Раздел 20 совместно с 15.42

16.14 Функция BioAPI_BSPDetach

16.14.1Данная функция определена в БиоАПИ следующим образом:

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

16.14.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **bspDetach** и тип сообщения ответа **bspDetach**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
BSPDetach-RequestParams ::= SEQUENCE {
                originalBSPHandle BioAPI-HANDLE
        }
```

и

```
BSPDetach-ResponseParams ::= NULL
```

16.14.3 Когда инфраструктура получает вызов к функции **BioAPI_BSPDetach** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (читай *originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.14.3.1 Если главная конечная точка является локальной конечной точкой, инфраструктура должна выполнить следующие действия в указанном порядке:

a) совершить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с такими же значениями параметра, как и во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования из *originalBSPHandle* согласно 15.42;

b) Если возвращенное значение внутреннего вызова не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

c) проверить таблицу **AttachSessionLocalReferences** (см. 18.8) на наличие поля, в котором значение компонента **originalBSPHandle** является *originalBSPHandle*;

d) Если такое поле не обнаружено, возвратить значение **BioAPIERR_NOT_A_RUNNING_BSP** локальному приложению без выполнения следующих действий;

e) удалить поле таблицы **AttachSessionLocalReferences** (выполняя действия, указанные в 18.8.3);

f) возвратить значение 0 локальному приложению.

16.14.3.2 Если главная конечная точка это второстепенная конечная точка инфраструктуры, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (читай *incomingRequestParams*) типа **BSPDetach-RequestParams** (см. 16.14.2) путем преобразования параметров вызова функции **BioAPI_BSPDetach** согласно 16.14.5;

b) создать и отправить сообщение запроса ПМО БиоАПИ **bspDetach** (см. 13.2) с ИИР второстепенной конечной точки, установленной на ИИР главной конечной точки, и со значением параметра, установленным на *incomingRequestParams*;

c) принять соответствующее сообщение ответа ПМО БиоАПИ **bspDetach** (см. 13.6);

d) Если возвращенное значение сообщения ответа ПМО БиоАПИ не является 0, возвратить такое значение локальному приложению без выполнения следующих действий;

e) проверить таблицу **AttachSessionLocalReferences** (см. 18.8) на наличие поля, в котором значением компонента **originalBSPHandle** является *originalBSPHandle*;

f) Если такое поле не обнаружено, возвратить значение **BioAPIERR_NOT_A_RUNNING_BSP** локальному приложению без выполнения следующих действий;

g) удалить поле таблицы **AttachSessionLocalReferences** (выполняют действия, указанные в 18.8.3);

h) возвратить значение 0 локальному приложению.

18.14.3.3 Если такое поле не обнаружено, отправить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

18.14.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) а **bspDetach** от главной конечной точки, она выполняет следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **BSPDetach-RequestParams** (см. 16.14.2) сообщения запроса ПМО БиоАПИ **bspDetach**;

b) совершить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPDetach**, в котором параметры вызова функции должны быть установлены путем преобразования из *incomingRequestParams* согласно 16.14.5;

c) Если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspDetach** (см. 13.3) с возвращаемым значением, установленным на такое значение, без выполнения следующих действий;

d) проверить таблицу **AttachSessionRemoteReferences** (см. 18.9) на наличие поля, в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** *incomingRequestParams*;

e) Если такое поле не обнаружено, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspDetach** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_NOT_A_RUNNING_BSP** без выполнения следующих действий;

f) удалить поле таблицы **AttachSessionRemoteReferences** (выполняют действия, указанные в 18.9.3);

g) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **bspDetach** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращаемым значением, установленным на 0.

16.14.5 Преобразование между параметрами функции Си **BioAPI_BSPDetach** и типом АСН.1 **BSPDetach-RequestParams** (см. 16.14.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 50.

Таблица 50 – Преобразования данных между параметрами функции **BioAPI_BSPDetach** и типом АСН.1 **BSPDetach-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26

16.15 Функция BioAPI_EnableEvents

16.15.1 Данная функция определена в БиоАПИ следующим образом:

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

16.15.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **enableEvents** и тип сообщения ответа **enableEvents**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
EnableEvents-RequestParams ::= SEQUENCE {
        originalBSPHandle    BioAPI-HANDLE,
        unitEvents           BioAPI-UNIT-EVENT-TYPE-MASK
    }
```

и

```
EnableEvents-ResponseParams ::= NULL
```

16.15.3 Когда инфраструктура получает вызов к функции **BioAPI_EnableEvents** от локального приложения, она должна сначала

определить главную конечную точку и исходный обработчик ПБУ (читай *originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к теми же функции с такими же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **enableUnitEvents** согласно разделу 27, выполняя действия, указанные в 16.15.5 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в данном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.15.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **enableUnitEvents** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_EnableEvents** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **enableUnitEvents** согласно разделу 28, выполняя действия, указанные в 16.19.5 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в данном разделе.

16.15.5 Преобразование между параметрами функции Си **BioAPI_EnableEvents** и типом АСН.1 **EnableUnitEvents-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 51.

Таблица 51 – Преобразование данных между параметрами функции **BioAPI_EnableEvents** и типом АСН.1 **EnableUnitEvents-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздел настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
Events	unitEvents	15.31

16.16 Функция BioAPI_EnableEventNotifications

16.16.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_EnableEventNotifications
        (const BioAPI_UUID *BSPUuid,
        BioAPI_EVENT_MASK Events);
```

16.16.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **EnableEventNotifications** и тип сообщения ответа **EnableEventNotifications**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
EnableEventNotifications-RequestParams ::= SEQUENCE {
                bspProductUuid      BioAPI-UUID,
                unitEventTypes          BioAPI-UNIT-EVENT-TYPE-MASK
        }
```

и

```
EnableEventNotifications-ResponseParams ::= NULL
```

16.16.3 Следующий тип АСН.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
EnableCallParams ::= SEQUENCE {
        bspUuid             BioAPI-UUID,
        unitEventTypes          BioAPI-UNIT-EVENT-TYPE-MASK
    }
```

16.16.4 Когда инфраструктура получает вызов к функции **BioAPI_EnableEventNotifications** от локального приложения, должна сначала определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.16.4.1 Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*enableCallParams*) типа **EnableCallParams** (см. 16.16.3) путем преобразования параметров вызова функции **BioAPI_EnableEventNotifications** согласно 16.16.6;

b) Если таблица **UnitEventNotificationDisablers** (см. 18.7) содержит поле, в котором компонент **referrerEndpointIRI** установлен на ИИР локальной точки, а компонент **bspProductUuid** установлен на *bspProductUuid*, удалить это поле;

c) Если компонент **unitEventTypes** *enableCallParams* указывает, что хоть один тип операции уведомления должен быть отключен, добавить поле к таблице **UnitEventNotificationDisablers** (см. 18.7), в котором:

1) компонент **referrerEndpointIRI** должен быть установлен на ИИР локальной конечной точки;

2) компонент **bspProductUuid** должен быть установлен на *bspProductUuid*; и

3) компонент **unitEventTypes** должен быть установлен на компонент **unitEventTypes** *enableCallParams*;

d) возвратить значение 0 локальному приложению.

16.16.4.2 Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*enableCallParams*) типа **EnableCallParams** (см. 16.16.3) путем преобразования параметров вызова функции **BioAPI_EnableEventNotifications** согласно 16.16.6;

b) создать временное абстрактное значение (*outgoingRequestParams*) типа **EnableEventNotifications-RequestParams** (см. 16.16.2), в котором:

1) компонент **bspProductUuid** должен быть установлен на *bspProductUuid*; и

2) компонент **unitEventTypes** должен быть установлен на компонент **unitEventTypes** *enableCallParams*;

c) создать и отправить сообщение запроса ПМО БиоАПИ **enableEventNotifications** (см. 13.2) с ИИР второстепенной конечной точки, установленной на ИИР главной конечной точки, со значением параметра, установленным на *outgoingRequestParams*;

d) принять соответствующее сообщение ответа ПМО БиоАПИ **enableEventNotifications** (см. 13.6);

e) Если возвращенное значение сообщения ответа ПМО БиоАПИ не равно 0, удалить поле, добавленное в начале таблицы **UnitEventNotificationDisablers**, и возвратить такое значение локальному приложению без выполнения следующих действий;

f) возвратить значение 0 локальному приложению.

16.16.4.3 Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.16.5 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **enableEventNotifications** от главной конечной точки, она выполняет следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **EnableEventNotifications-RequestParams** (см. 16.16.2) сообщения запроса ПМО БиоАПИ **enableEventNotifications**;

b) в случае, если таблица **UnitEventNotificationDisablers** (см. 18.7) содержит поле, в котором компонент **referrerEndpointIRI** установлен на ИИР главной конечной точки, а компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** *incomingRequestParams*, удалить это поле;

c) в случае, сли компонент **unitEventTypes** *incomingRequestParams* указывает, что хоть один тип операции уведомления должен быть отключен, добавить поле к таблице **UnitEventNotificationDisablers** (см. 18.7), в котором:

1) компонент **referrerEndpointIRI** должен быть установлен на ИИР главной конечной точки;

2) компонент **bspProductUuid** должен быть установлен из компонента **bspProductUuid** *incomingRequestParams*;

3) компонент **unitEventTypes** должен быть установлен из компонента **unitEventTypes** *incomingRequestParams*;

d) создать и отправить соответствующее сообщение запроса ПМО БиоАПИ **enableEventNotifications** (см. 18.3) с возвращаемым значением, установленным на 0.

16.16.6 Преобразование между параметрами функции Си **BioAPI_EnableEventNotifications** и типом АСН.1 **EnableCallParams** (см. 16.16.3) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 52.

Таблица 52 – Преобразование данных между параметрами функции **BioAPI_EnableEventNotifications** и типом АСН.1 **EnableCallParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздел настоящего стандарта
BSPUuid	bspUuid	Раздел 19 совместно с 15.58
Events	unitEventTypes	15.31

16.17 Функция BioAPI_ControlUnit

16.17.1 Данная функция определена в БиоАПИ следующим образом:

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

16.17.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **controlUnit** и тип сообщения ответа **controlUnit**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
ControlUnit-RequestParams ::= SEQUENCE {
                originalBSPHandle BioAPI-HANDLE,
                unitID BioAPI-UNIT-ID,
                controlCode UnsignedInt,
                inputData BioAPI-DATA
        }
```

и

```
ControlUnit-ResponseParams ::= SEQUENCE {
                outputData BioAPI-DATA
        }
```

16.17.3 Когда инфраструктура получает вызов к функции **BioAPI_ControlUnit** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (читай *originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если второстепенной конечной точкой является главная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с

главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **controlUnit** согласно разделу 27, выполняя действия, указанные в 16.17.5 и 16.17.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в данном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.17.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **controlUnit** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_ControlUnit** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **controlUnit** согласно разделу 28, выполняя действия, указанные в 16.17.5 и 16.17.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в данном разделе.

16.17.5 Преобразование между параметрами функции Си **BioAPI_ControlUnit** и типом АСН.1 **ControlUnit-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно Таблице 53.

Таблица 53 – Преобразования данных между параметрами функции **BioAPI_ControlUnit** и типом АСН.1 **ControlUnit-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
UnitID	unitID	15.55
ControlCode	controlCode	15.1.5
InputData	inputData	Раздел 19 совместно с 15.22
OutputData	Отсутствует	Раздел 22

16.17.6 Преобразование между параметрами функции Си **BioAPI_ControlUnit** и типом АСН.1 **ControlUnit-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 54.

Таблица 54 – Преобразования данных между параметрами функции **BioAPI_ControlUnit** и типом АСН.1 **ControlUnit-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
OutputData	outputData	Раздел 20 совместно с 15.22

16.18 Функция BioAPI_Control

16.18.1 Данная функция определена в БиоАПИ следующим образом:

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

16.18.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **control** и тип сообщения ответа **control**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
Control-RequestParams ::= SEQUENCE {
        original        BSPHandle BioAPI-HANDLE,
        unitID          BioAPI-UNIT-ID,
        controlCode     BioAPI-UUID,
        inputData       BioAPI-DATA
    }
```

и

```
Control-ResponseParams ::= SEQUENCE {
        outputData      BioAPI-DATA
    }
```

16.18.3 Когда инфраструктура получает вызов к функции **BioAPI_Control** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (читай *originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если локальной конечной точкой является главная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями контроля запроса/ответа ПМО БиоАПИ как указано в разделе 27, выполняя действия, указанные в 16.18.5 и 16.18.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в данном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.18.4 Когда инфраструктура получает сообщение запроса контроля ПМО БиоАПИ (см. 13.9) от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Control** для создания и отправления соответствующего сообщения ответа контроля ПМО БиоАПИ согласно 13.10, выполняя действия, указанные в 16.18.5 и 16.18.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в данном разделе.

16.18.5 Преобразование между параметрами функции Си **BioAPI_Control** и типом АСН.1 **Control-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 55.

Таблица 55 – Преобразование данных между параметрами функции **BioAPI_Control** и типом АСН.1 **Control-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
UnitID	unitID	15.55
ControlCode	controlCode	15.1.5
InputData	inputData	Раздел 19 совместно с 15.22
OutputData	Отсутствует	Раздел 22

16.18.6 Преобразование между параметрами функции Си **BioAPI_Control** и типом АСН.1 **Control-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 56.

Таблица 56 – Преобразование данных между параметрами функции **BioAPI_Control** и типом АСН.1 **Control- Control-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздел настоящего стандарта
OutputData	outputData	Раздел 20 совместно с 15.22

16.19 Функция BioAPI_FreeBIRHandle

16.19.1 Данная функция определена в БиоАПИ следующим образом:

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

16.19.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **freeBIRHandle** и тип сообщения ответа **freeBIRHandle**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
FreeBIRHandle-RequestParams ::= SEQUENCE {
        original                BSPHandle BioAPI-HANDLE,
        birHandle       BioAPI-BIR-HANDLE
}
```

и

```
FreeBIRHandle-ResponseParams ::= NULL
```

16.19.3 Когда инфраструктура получает вызов к функции **BioAPI_FreeBIRHandle** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (читай *originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечня точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **freeBIRHandle** согласно разделу 27, выполняя действия, указанные в 16.19.5 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.19.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **freeBIRHandle** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к

BioAPI_FreeBIRHandle для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **freeBIRHandle** согласно разделу 28, выполняя действия, указанные в 16.19.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.19.5 Преобразование между параметрами функции Си **BioAPI_FreeBIRHandle** и типом АСН.1 **FreeBIRHandle-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 57.

Таблица 57 – Преобразования данных между параметрами функции **BioAPI_FreeBIRHandle** и типом АСН.1 **FreeBIRHandle-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздал настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
Handle	birHandle	16.12

16.20 Функция BioAPI_GetBIRFromHandle

16.20.1 Данная функция определена в БиоАПИ следующим образом:

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

16.20.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **getBIRFromHandle** и тип сообщения ответа **getBIRFromHandle**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
GetBIRFromHandle-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                birHandle BioAPI-    BIR-HANDLE
        }
```

и

```
GetBIRFromHandle-ResponseParams ::= SEQUENCE {
        bir                     BioAPI-BIR
}
```

16.20.3 Когда инфраструктура получает вызов к функции **BioAPI_GetBIRFromHandle** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точкая инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **getBIRFromHandle** согласно разделу 27, выполняя действия, указанные в 16.20.5 и 16.20.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.20.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **getBIRFromHandle** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_GetBIRFromHandle** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **getBIRFromHandle** согласно разделу 28, выполняя действия, указанные в 16.20.5 и 16.20.6, для преобразования между

параметрами функции и компонентами ACH.1, если это установлено в указанном разделе.

16.20.5 Преобразование между параметрами функции Си **BioAPI_GetBIRFromHandle** и типом ACH.1 **GetBIRFromHandle-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 58.

Таблица 58 – Преобразования данных между параметрами функции **BioAPI_GetBIRFromHandle** и типом ACH.1 **GetBIRFromHandle-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, подраздел настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
Handle	birHandle	15.12
BIR	Отсутствует	Раздел 22

16.20.6 Преобразование между параметрами функции Си **BioAPI_GetBIRFromHandle** и типом ACH.1 **GetBIRFromHandle-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 59.

Таблица 59 – Преобразование данных между параметрами функции **BioAPI_GetBIRFromHandle** и типом ACH.1 **GetBIRFromHandle-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, подраздел настоящего стандарта
BIR	bir	Раздел 20 совместно с 15.6

16.21 Функция BioAPI_GetHeaderFromHandle

16.21.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **getHeaderFromHandle** и тип сообщения ответа **getHeaderFromHandle**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
GetHeaderFromHandle-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        birHandle               BioAPI-BIR-HANDLE
    }
```

и

```
GetHeaderFromHandle-ResponseParams ::= SEQUENCE {
        header                  BioAPI-BIR-HEADER
    }
```

16.21.3 Когда инфраструктура получает вызов к функции **BioAPI_GetHeaderFromHandle** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с такими же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **getHeaderFromHandle** согласно разделу 27, выполняя действия, указанные в 16.21.5 и 16.21.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная

конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.21.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **getHeaderFromHandle** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_GetHeaderFromHandle** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **getHeaderFromHandle** согласно разделу 28, выполняя действия, указанные в 16.21.5 и 16.21.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.21.5 Преобразование между параметрами функции Си **BioAPI_GetHeaderFromHandle** и типом АСН.1 **GetHeaderFromHandle-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 60.

Таблица 60 – Преобразования данных между параметрами функции **BioAPI_GetHeaderFromHandle** и типом АСН.1 **GetHeaderFromHandle-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздел настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
Handle	birHandle	15.12
Header	Отсутствует	Раздел 22

16.21.6 Преобразование между параметрами функции Си **BioAPI_GetHeaderFromHandle** и типом АСН.1 **GetHeaderFromHandle-ResponseParams** выполняют путем преобразования между

индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 61.

Таблица 61 – Преобразование данных между параметрами функции **BioAPI_GetHeaderFromHandle** и типом АСН.1 **GetHeaderFromHandle-ResponseParams**

Параметр функции	Компонент типа АСН.1	Ссылки
Header	header	Раздел 20 совместно с 15.13

16.22 Функция BioAPI_SubscribeToGUIEvents

16.22.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_SubscribeToGUIEvents
        (const BioAPI_UUID *GUIEventSubscriptionUuid,
        const BioAPI_UUID *BSPUuid,
        const BioAPI_HANDLE *BSPHandle,
        BioAPI_GUI_SELECT_EVENT_HANDLER GUISelectEventHandler,
        void *GUISelectEventHandlerCtx,
        BioAPI_GUI_STATE_EVENT_HANDLER GUIStateEventHandler,
        void *GUIStateEventHandlerCtx,
        BioAPI_GUI_PROGRESS_EVENT_HANDLER GUIProgressEventHandler,
        void *GUIProgressEventHandlerCtx);
```

16.22.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **subscribeToGUIEvents** и тип сообщения ответа **subscribeToGUIEvents**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
SubscribeToGUIEvents-RequestParams ::= SEQUENCE {
        guiEventSubscriptionUuid        BioAPI-UUID OPTIONAL,
        bspProductUuid                  BioAPI-UUID OPTIONAL,
        originalBSPHandle               BioAPI-HANDLE OPTIONAL,
        guiSelectEventSubscribed        BOOLEAN,
```

```
        guiStateEventSubscribed          BOOLEAN,
        guiProgressEventSubscribed       BOOLEAN
    }
```

и

```
SubscribeToGUIEvents-ResponseParams ::= NULL
```

16.22.3 Следующий тип АСН.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
SubscribeToGUIEventsCallParams ::= SEQUENCE {
        guiEventSubscriptionUuid          BioAPI-UUID OPTIONAL,
        bspUuid BioAPI-UUID               OPTIONAL,
        bspHandle BioAPI-HANDLE           OPTIONAL,
        guiSelectEventHandlerAddress      MemoryAddress,
        guiSelectEventHandlerContext      MemoryAddress,
        guiStateEventHandlerAddress       MemoryAddress,
        guiStateEventHandlerContext       MemoryAddress,
        guiProgressEventHandlerAddress    MemoryAddress,
        guiProgressEventHandlerContext    MemoryAddress
    }
```

16.22.4 Когда инфраструктура принимает вызов к функции **BioAPI_SubscribeToGUIEvents** от локального приложения, она сначала должна определить:

a) главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23 (в случае, если параметр **BSPUuid** имеет значение, отличающееся от **NULL**) или

b) главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из переменной Си, выделенной параметром **BSPHandle** согласно разделу 24 (в случае, если параметр **BSPHandle** имеет значение, отличающееся от **NULL**),

а затем выполняют действия, указанные в одном из следующих подпунктов.

Примечание – Необходимо, чтобы только один из двух указанных параметров имел значение **NULL** (см. ИСО/МЭК 19784-1, 8.3.8.1).

16.22.4.1 Если главной конечной точкой является локальная конечная точка, инфраструктура выполняет следующие действия в указанном порядке:

a) совершить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как и во входящем вызове, за исключением:

1) в случае, если параметр **BSPUuid** входящего вызова имел значение, отличающееся от **NULL**, параметр **BSPUuid** должен быть установлен путем преобразования из *bspProductUuid*, согласно разделу 18 совместно с 15.58 и

2) в случае, если параметр **BSPHandle** входящего вызова имеет значение, отличающееся от **NULL**, параметр **BSPHandle** должен быть установлен путем преобразования из *originalBSPHandle* согласно разделу 19 совместно с 15.42;

b) Если возвращенное значение внутреннего вызова не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*subscribeToGUIEventsCallParams*) типа **SubscribeToGUIEventsCallParams** (см. 16.22.3) путем преобразования из параметров вызова функции **BioAPI_SubscribeToGUIEvents** согласно 16.22.6;

d) добавить поле в таблицу **GUIEventLocalSubscriptions** (см. 18.10), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** *subscribeToGUIEventsCallParams* (присутствие и значение);

2) компонент **hostingEndpointIRI** должен быть установлен на ИИР локальной точки;

3) в случае, если необязательный компонент **bspUuid** *subscribeToGUIEventsCallParams* присутствует, компонент **bspProductUuid** поля должен быть установлен на *bspProductUuid*; в противном случае он должен быть установлен из компонента **bspProductUuid** поля таблицы **AttachSessionLocalReferences** (см. 18.8), в котором компонент **originalBSPHandle** имеет значение *originalBSPHandle*;

4) в случае, если необязательный компонент **bspUuid** *subscribeToGUIEventsCallParams* отсутствует или имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

5) в случае, если необязательный компонент **bspHandle** *subscribeToGUIEventsCallParams* присутствует, необязательный компонент **originalBSPHandle** должен быть установлен на *originalBSPHandle*; в противном случае, он должен отсутствовать и

6) компоненты **guiSelectEventHandlerAddress**, **guiSelectEventHandlerContext**, **guiStateEventHandlerAddress**, **guiStateEventHandlerContext**, **guiProgressEventHandlerAddress** и **guiProgressEventHandlerContext** должны быть установлены из компонентов *subscribeToGUIEventsCallParams* с аналогичными же именами;

е) возвратить значение 0 локальному приложению.

16.22.4.2 Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна выполнять следующие действия в указанном порядке:

а) создать временное абстрактное значение (*subscribeToGUIEventsCallParams*) типа **SubscribeToGUIEventsCallParams** (см. 16.22.3) путем

преобразования из параметров **BioAPI_SubscribeToGUIEvents** вызова функции согласно 16.22.6;

b) создать временное абстрактное значение (читай *outgoingRequestParams*) типа **SubscribeToGUIEvents-RequestParams** (см. 16.22.2), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** *subscribeToGUIEventsCallParams* (присутствие и значение);

2) в случае, если необязательный компонент **bspUuid** *subscribeToGUIEventsCallParams* присутствует, компонент **bspProductUuid** *outgoingRequestParams* должен быть установлен на *bspProductUuid*; в противном случае, он должен отсутствовать;

3) в случае, если необязательный компонент **bspHandle** *subscribeToGUIEventsCallParams* присутствует, необязательный компонент **originalBSPHandle** *outgoingRequestParams* должен присутствовать и также должен быть установлен на *originalBSPHandle*; в противном случае, он должен отсутствовать;

4) в случае, если компонент **guiSelectEventHandlerAddress** *subscribeToGUIEventsCallParams* имеет значение, отличающееся от 0, компонент **guiSelectEventSubscribed** *outgoingRequestParams* должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

5) в случае, если компонент **guiStateEventHandlerAddress** *subscribeToGUIEventsCallParams* имеет значение, отличающееся от 0, компонент **guiStateEventSubscribed** *outgoingRequestParams* должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**; и

6) в случае, если компонент **guiProgressEventHandlerAddress** *subscribeToGUIEventsCallParams* имеет значение, отличающееся от 0, компонент **guiProgressEventSubscribed** *outgoingRequestParams* должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

c) создать и отправить сообщение запроса ПМО БиоАПИ **subscribeToGUIEvents** (см. 13.2) с ИИР второстепенной конечной точки, установленным на ИИР главной конечной точки, и значением параметра, установленным на *outgoingRequestParams*;

d) принять соответствующее сообщение ответа ПМО БиоАПИ **subscribeToGUIEvents** (см. 13.6);

e) в случае, если возвращенное значение сообщения ответа ПМО БиоАПИ не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

f) добавить поле в таблицу **GUIEventLocalSubscriptions** (см. 18.10), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** *subscribeToGUIEventsCallParams* (присутствие и значение);

2) компонент **hostingEndpointIRI** должен быть установлен на ИИР главной конечной точки;

3) в случае, если необязательный компонент **bspUuid** *subscribeToGUIEventsCallParams* присутствует, компонент **bspProductUuid** поля устанавливается на *bspProductUuid*; в противном случае, он должен быть установлен из компонента **bspProductUuid** поля таблицы **AttachSessionLocalReferences** (см. 18.8), в котором компонент **originalBSPHandle** имеет значение *originalBSPHandle*;

4) Если необязательный компонент **bspUuid** *subscribeToGUIEventsCallParams* отсутствует или имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля должен быть установлен на **TRUE**; в противном случае, он должен быть установлен на **FALSE**;

5) Если необязательный компонент **bspHandle** *subscribeToGUIEventsCallParams* присутствует, необязательный компонент **originalBSPHandle** поля должен быть установлен на *originalBSPHandle*; в противном случае, он должен отсутствовать и

6) компоненты **guiSelectEventHandlerAddress**, **guiSelect-EventHandlerContext**, **guiStateEventHandlerAddress**, **guiStateEventHandlerContext**, **guiProgressEventHandlerAd-dress**, и **guiProgressEventHandlerContext** должны быть установлены из компонентов *subscribeToGUIEventsCallParams* с такими же именами;

g) возвратить значение 0 локальному приложению.

16.22.4.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.22.5 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **subscribeToGUIEvents** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **SubscribeToGUIEvents-RequestParams** (см. 16.22.2) сообщения запроса ПМО БиоАПИ **subscribeToGUIEvents**;

b) создать временное абстрактное значение (*subscribeToGUIEventsCallParams*) типа **SubscribeToGUIEventsCallParams** (см. 16.22.3), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента

guiEventSubscriptionUuid *incomingRequestParams* (присутствие и значение);

2) Если необязательный компонент **bspProductUuid** *incomingRequestParams* присутствует, компонент **bspUuid** *subscribeToGUIEventsCallParams* должен быть установлен из такого компонента; в противном случае он должен отсутствовать;

3) Если необязательный компонент **originalBSPHandle** *incomingRequestParams* присутствует, компонент **bspHandle** *subscribeToGUIEventsCallParams* должен быть установлен из такого компонента; в противном случае он должен отсутствовать;

4) Если компонент **guiSelectEventSubscribed** *incomingRequestParams* имеет значение **TRUE**, компонент **guiSelectEventHandlerAddress** должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0; в противном случае он должен быть установлен на 0.

Примечание 1 – Действительный адрес памяти, определенный для **guiSelectEventHandlerAddress**, не имеет значения, так как никакие вызовы к такому адресу совершаться не будут (см. 17.2.5);

5) Если компонент **guiStateEventSubscribed** *incomingRequestParams* имеет значение **TRUE**, компонент **guiStateEventHandlerAddress** должен быть установлен в определенный реализацией адрес памяти отличный от 0; в противном случае, он должен быть установлен на 0.

Примечание 2 – Действительный адрес памяти, определенный для **guiStateEventHandlerAddress** не имеет значения, так как никакие вызовы к такому адресу выполняться не должны (см. 17.3.5);

6) Если компонент **guiProgressEventSubscribed** *incomingRequestParams* имеет значение **TRUE**, компонент **guiProgressEventHandlerAddress** должен быть установлен в

определенный реализацией адрес памяти, отличающийся от 0; в противном случае, он должен быть установлен на 0 и

Примечание 3 – Действительный адрес памяти, определенный для **guiProgressEventHandlerAddress** не имеет значения, так как никакие вызовы к такому адресу выполняться не должны (см. 17.4.5);

7) компоненты **guiSelectEventHandlerContext**, **guiStateEvent-HandlerContext**, and **guiProgressEventHandlerContext** должны быть установлены на 0;

c) в случае, если оба компонента **bspProductUuid** и **originalBSPHandle** *incomingRequestParams* присутствуют, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **subscribeToGUIEvents** (см. 17.3) с возвращаемым значением, установленным на **BioAPIERR_UUID_AND_HANDLE_BOTH_PRESENT** без выполнения следующих действий;

d) в случае, если оба компонента **bspProductUuid** и **originalBSPHandle** *incomingRequestParams* отсутствуют, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **subscribeToGUIEvents** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_UUID_AND_HANDLE_BOTH_ABSENT** без выполнения следующих действий;

e) совершить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_SubscribeToGUIEvents**, в котором параметры вызова функции должны быть установлены путем преобразования *subscribeToGUIEventsCallParams* согласно 16.22.6;

f) Если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **subscribeToGUIEvents** (см. 13.3) с возвращаемым значением, установленным на это значение, без выполнения следующих действий;

g) добавить поле в таблицу **GUIEventRemoteSubscriptions** (см. 18.11), в котором:

1) компонент **subscriberEndpointIRI** должен быть установлен на ИИР главной конечной точки;

2) необязательный компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** *incomingRequestParams* (присутствие и значение);

3) Если необязательный компонент **bspProductUuid** *incomingRequestParams* присутствует, компонент **bspProductUuid** поля должен быть установлен из такого компонента; в противном случае, он должен быть установлен из компонента **bspProductUuid** поля таблицы **AttachSessionRemoteReferences** (см. 18.9), в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** *incomingRequestParams*;

4) Если необязательный компонент **originalBSPHandle** *incomingRequestParams* присутствует, компонент **bspHandle** поля должен быть установлен из такого компонента; в противном случае он должен отсутствовать; и

5) компоненты **guiSelectEventSubscribed**, **guiStateEventSubscribed** и **guiProgressEventSubscribed** должны быть установлены из *incomingRequestParams* с такими же именами;

h) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **subscribeToGUIEvents** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращенным значением, установленным на 0.

15.22.6 Преобразование между параметрами функции Си **BioAPI_SubscribeToGUIEvents** и типом АСН.1 **SubscribeToGUIEventsCallParams** (см. 16.22.3) выполняют путем

преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 62.

Таблица 62 – Преобразование данных между параметрами функции **BioAPI_SubscribeToGUIEvents** и типом АСН.1 **SubscribeToGUIEvents-CallParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздел, пункт настоящего стандарта
GUIEventSubscriptionUuid	guiEventSubscriptionUuid	Раздел 19 совместно с 15.58
BSPUuid	bspUuid	Раздел 19 совместно с 15.58
BSPHandle	bspHandle	Раздел 19 совместно с 15.42
GUISelectEventHandler	guiSelectEventHandlerAddress	15.1.7
GUISelectEventHandler	guiSelectEventHandlerAddress	15.1.7
GUIStateEventHandler	guiStateEventHandlerAddress	15.1.7
GUIStateEventHandler	guiStateEventHandlerAddress	15.1.7
GUIProgressEventHandler	guiProgressEventHandlerAddress	15.1.7
GUIProgressEventHandlerCtx	guiProgressEventHandlerContext	15.1.7

16.23 Функция BioAPI_UnsubscribeFromGUIEvents

16.23.1 Данная функция определена в БиоАПИ следующим образом:

BioAPI_RETURN BioAPI BioAPI_UnsubscribeFromGUIEvents

```
(const BioAPI_UUID *GUIEventSubscriptionUuid,
const BioAPI_UUID *BSPUuid,
const BioAPI_HANDLE *BSPHandle,
BioAPI_GUI_SELECT_EVENT_HANDLER GUISelectEventHandler,
void *GUISelectEventHandlerCtx,
BioAPI_GUI_STATE_EVENT_HANDLER GUIStateEventHandler,
void *GUIStateEventHandlerCtx,
BioAPI_GUI_PROGRESS_EVENT_HANDLER GUIProgressEventHandler,
void *GUIProgressEventHandlerCtx);
```

16.23.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **unsubscribeFromGUIEvents** и тип сообщения ответа **unsubscribeFromGUIEvents**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
UnsubscribeFromGUIEvents-RequestParams ::= SEQUENCE {
        guiEventSubscriptionUuid        BioAPI-UUID OPTIONAL,
        bspProductUuid                  BioAPI-UUID OPTIONAL,
        originalBSPHandle               BioAPI-HANDLE OPTIONAL,
        guiSelectEventSubscribed        BOOLEAN,
        guiStateEventSubscribed         BOOLEAN,
        guiProgressEventSubscribed      BOOLEAN
    }
```

и

```
UnsubscribeFromGUIEvents-ResponseParams ::= NULL
```

16.23.3 Следующий тип АСН.1 применяют при спецификации поведения инфраструктуры, но его абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
UnsubscribeFromGUIEventsCallParams ::= SEQUENCE {
        guiEventSubscriptionUuid        BioAPI-UUID OPTIONAL,
        bspUuid BioAPI-UUID             OPTIONAL,
        bspHandle BioAPI-HANDLE         OPTIONAL,
        guiSelectEventHandlerAddress    MemoryAddress,
        guiSelectEventHandlerContext    MemoryAddress,
        guiStateEventHandlerAddress     MemoryAddress,
        guiStateEventHandlerContext     MemoryAddress,
        guiProgressEventHandlerAddress  MemoryAddress,
```

guiProgressEventHandlerContext MemoryAddress

}

16.23.4 Когда инфраструктура принимает вызов к функции **BioAPI_UnsubscribeFromGUIEvents** от локального приложения, она сначала должна определить:

a) главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23 в случае, если параметр **BSPUuid** имеет значение, отличающееся от **NULL** или

b) главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из переменной Си, выделенной параметром **BSPHandle** согласно разделу 24 в случае, если параметр **BSPHandle** имеет значение, отличающееся от **NULL**;

а затем выполняют действия, указанные в одном из следующих подпунктов.

Примечание – Необходимо, чтобы только один из двух указанных параметров имел значение **NULL** (см. ИСО/МЭК 19784-1, 8.3.8.1).

16.23.4.1 Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить следующие действия в указанном порядке:

a) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как и во входящем вызове, за исключением следующего:

1) в случае, если параметр **BSPUuid** входящего вызова имел значение, отличающееся от **NULL**, параметр **BSPUuid** должен быть установлен путем преобразования из *bspProductUuid* согласно разделу 19 совместно с 15.58 и

2) в случае, если параметр **BSPHandle** входящего вызова имеет значение, отличающееся от **NULL**, параметр **BSPHandle** должен быть установлен путем преобразования из *originalBSPHandle* согласно разделу 19 совместно с 15.42;

b) в случае, если возвращенное значение внутреннего вызова не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*unsubscribeFromGUIEventsCallParams*) типа **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3) путем преобразования из параметров вызова функции **BioAPI_UnsubscribeFromGUIEvents** согласно 16.22.6;

d) проверить таблицу **GUIEventLocalSubscriptions** (см. 18.10) на наличие поля, в котором:

1) необязательный компонент **guiEventSubscriptionUuid** имеет такое же присутствие и значение, как и необязательный компонент **guiEventSubscriptionUuid** *unsubscribeFromGUIEventsCallParams*;

2) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки;

3) в случае, если необязательный компонент **bspUuid** *unsubscribeFromGUIEventsCallParams* присутствует, компонент **bspProductUuid** поля имеет значение *bspProductUuid*; в противном случае он имеет такое же значение, как и компонент **bspProductUuid** поля таблицы **AttachSessionLocalReferences** (см. 18.8), в котором компонент **originalBSPHandle** имеет значение *originalBSPHandle*;

4) в случае, если необязательный компонент **bspUuid** *unsubscribeFromGUIEventsCallParams* отсутствует или имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля имеет значение **TRUE**; в противном случае он имеет значение **FALSE**;

5) в случае, если необязательный компонент **bspHandle** *unsubscribeFromGUIEventsCallParams* присутствует, необязательный компонент **originalBSPHandle** поля также присутствует и имеет значение *originalBSPHandle*; в противном случае компонент отсутствует и

6) компоненты **guiSelectEventHandlerAddress**, **guiSelectEventHandlerContext**, **guiStateEventHandlerAddress**, **guiStateEventHandlerContext**, **guiProgressEventHandlerAddress** и **guiProgressEventHandlerContext** имеют те же значения, как и компоненты *unsubscribeFromGUIEventsCallParams* с такими же именами;

e) в случае, если такое поле не обнаружено, возвратить значение **BioAPIERR_NO_SUCH_SUBSCRIPTION_FOUND** локальному приложению без выполнения следующих действий;

f) удалить поле таблицы **GUIEventLocalSubscriptions** (применяется 18.10.3);

g) возвратить значение 0 локальному приложению.

16.23.4.2 Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*unsubscribeFromGUIEventsCallParams*) типа **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3) путем преобразования из параметров вызова функции **BioAPI_UnsubscribeFromGUIEvents** согласно 16.23.6;

b) создать временное абстрактное значение (*outgoingRequestParams*) типа **UnsubscribeFromGUIEvents-RequestParams** (см. 16.23.2), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** *unsubscribeFromGUIEventsCallParams* (присутствие и значение);

2) Если необязательный компонент **bspUuid** *unsubscribeFromGUIEventsCallParams* присутствует, компонент

bspProductUuid *outgoingRequestParams* должен быть установлен в *bspProductUuid*; в противном случае он должен отсутствовать;

3) Если необязательный компонент **bspHandle** *unsubscribeFromGUIEventsCallParams* присутствует, необязательный компонент **originalBSPHandle** *outgoingRequestParams* должен быть установлен в *originalBSPHandle*; в противном случае он должен отсутствовать;

4) Если компонент **guiSelectEventHandlerAddress** *unsubscribeFromGUIEventsCallParams* имеет значение, отличающееся от 0, компонент **guiSelectEventSubscribed** *outgoingRequestParams* должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

5) Если компонент **guiStateEventHandlerAddress** *unsubscribeFromGUIEventsCallParams* имеет значение, отличающееся от 0, компонент **guiStateEventSubscribed** *outgoingRequestParams* должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE** и

6) Если компонент **guiProgressEventHandlerAddress** *unsubscribeFromGUIEventsCallParams* имеет значение, отличающееся от 0, компонент **guiProgressEventSubscribed** *outgoingRequestParams* должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

c) создать и отправить сообщение запроса ПМО БиоАПИ **unsubscribeFromGUIEvents** (см. 13.2) с ИИР второстепенной конечной точки, установленным на ИИР главной конечной точки, со значением параметра, установленным на *outgoingRequestParams*;

d) принять соответствующее сообщение ответа ПМО БиоАПИ **unsubscribeFromGUIEvents** (см. 13.6);

e) Если возвращенное значение сообщения ответа ПМО БиоАПИ не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

f) проверить таблицу **GUIEventLocalSubscriptions** (см. 18.10) на наличие поля, в котором:

1) необязательный компонент **guiEventSubscriptionUuid** имеет такое же значение, как и необязательный компонент **guiEventSubscriptionUuid** *unsubscribeFromGUIEventsCallParams*;

2) компонент **hostingEndpointIRI** содержит ИИР главной конечной точки;

3) Если необязательный компонент **bspUuid** *unsubscribeFromGUIEventsCallParams* присутствует, необязательный компонент **bspProductUuid** поля имеет значение *bspProductUuid*; в противном случае он имеет такое же значение, как и компонент **bspProductUuid** поля таблицы **AttachSessionLocalReferences** (см. 18.8), в котором компонент **originalBSPHandle** имеет значение *originalBSPHandle*;

4) Если необязательный компонент **bspUuid** *unsubscribeFromGUIEventsCallParams* отсутствует или имеет значение, отличающееся от *bspProductUuid*, компонент **useBSPAccessUuid** поля имеет значение **TRUE**; в противном случае он имеет значение **FALSE**;

5) Если необязательный компонент **bspHandle** *unsubscribeFromGUIEventsCallParams* присутствует, необязательный компонент **originalBSPHandle** поля также присутствует и имеет значение *originalBSPHandle*; в противном случае отсутствует и

6) компоненты **guiSelectEventHandlerAddress**, **guiSelectEventHandlerContext**, **guiStateEventHandlerAddress**, **guiStateEventHandlerContext**, **guiProgressEventHandlerAddress**,

и **guiProgressEventHandlerContext** имеют те же значения, как и компоненты *unsubscribeFromGUIEventsCallParams* с такими же именами;

g) Если такое поле не обнаружено, возвратить значение **BioAPIERR_NO_SUCH_SUBSCRIPTION_FOUND** локальному приложению без выполнения следующих действий;

h) удалить поле таблицы **GUIEventLocalSubscriptions** (применяют 18.10.3);

i) возвратить значение 0 локальному приложению.

16.23.4.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.23.5 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **unsubscribeFromGUIEvents** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **UnsubscribeFromGUIEvents- RequestParams** (см. 16.23.2) сообщения запроса ПМО БиоАПИ **unsubscribeFromGUIEvents**;

b) создать временное абстрактное значение (*unsubscribeFromGUIEventsCallParams*) типа **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** *incomingRequestParams* (присутствие и значение);

2) Если необязательный компонент **bspProductUuid** *incomingRequestParams* присутствует, необязательный компонент **bspUuid** должен быть установлен из этого компонента; в противном случае он должен отсутствовать;

3) необязательный компонент **bspHandle** должен быть установлен из необязательного компонента **originalBSPHandle** *incomingRequestParams* (присутствие и значение);

4) Если компонент **guiSelectEventSubscribed** *incomingRequestParams* имеет значение **TRUE**, компонент **guiSelectEventHandlerAddress** *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти отличающийся от 0, который должен соответствовать указанному в 16.22.5, перечисление b); в противном случае он должен быть установлен на 0;

5) Если компонент **guiStateEventSubscribed** *incomingRequestParams* имеет значение **TRUE**, компонент **guiStateEventHandlerAddress** *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать указанному в 16.22.5, перечисление b); в противном случае он должен быть установлен на 0;

6) Если компонент **guiProgressEventSubscribed** *incomingRequestParams* имеет значение**TRUE**, компонент **guiProgressEventHandlerAddress** *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать указанному в 16.22.5, перечисление b); в противном случае он должен быть установлен на 0 и

7) компоненты **guiSelectEventHandlerContext**, **guiStateEventHandlerContext** и **guiProgressEventHandlerContext** должны быть установлены на 0;

c) Если оба компонента **bspProductUuid** и **originalBSPHandle** *incomingRequestParams* присутствуют, создать и отправить

соответствующее сообщение ответа ПМО БиоАПИ **unsubscribeFromGUIEvents** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_UUID_AND_HANDLE_BOTH_PRESENT** без выполнения следующих действий;

d) Если оба компонента **bspProductUuid** и **originalBSPHandle** *incomingRequestParams* отсутствуют, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unsubscribeFromGUIEvents** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_UUID_AND_HANDLE_BOTH_ABSENT** без выполнения следующих действий;

e) совершить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnsubscribeFromGUIEvents**, в котором параметры вызова функции должны быть установлены путем преобразования *unsubscribeFromGUIEventsCallParams* согласно 16.22.6;

f) Если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unsubscribeFromGUIEvents** (см. 13.3) с возвращаемым значением, установленным на это значение без выполнения следующих действий;

g) проверить таблицу **GUIEventRemoteSubscriptions** (см. 18.11) на наличие поля, в котором:

1) компонент **subscriberEndpointIRI** содержит ИИР главной конечной точки;

2) необязательный компонент **guiEventSubscriptionUuid** имеет такое же значение, как и необязательный компонент **guiEventSubscriptionUuid** *incomingRequestParams*;

3) в случае, если необязательный компонент **bspProductUuid** *incomingRequestParams* присутствует, компонент **bspProductUuid** поля

имеет такое же значение, как и указанный компонент; в противном случае, он имеет такое же значение, как и компонент **bspProductUuid** поля таблицы **AttachSessionRemoteReferences** (см. 18.9), в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** *incomingRequestParams*;

4) в случае, если необязательный компонент **originalBSPHandle** *incomingRequestParams* присутствует, компонент **bspHandle** поля имеет такое же значение, как и указанный компонент; в противном случае он отсутствует и

5) компоненты **guiSelectEventSubscribe**, **guiStateEventSubscribe** и **guiProgressEventSubscribed** имеют те же значения, как и компоненты *incomingRequestParams* с такими же именами;

h) Если такое поле не обнаружено, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unsubscribeFromGUIEvents** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_NO_SUCH_SUBSCRIPTION_FOUND** без выполнения следующих действий;

i) удалить поле таблицы **GUIEventRemoteSubscriptions** (выполняют действия, указанные в 18.11.3);

j) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unsubscribeFromGUIEvents** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращаемым значением, установленным на 0.

16.23.6 Преобразование между параметрами функции Си **BioAPI_UnsubscribeFromGUIEvents** и типом ACH.1 **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3) выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 63.

Таблица 63 –Преобразование данных между параметрами функции **BioAPI_UnsubscribeFromGUIEvents** и типом ACH.1 **UnsubscribeFromGUIEventsCallParams**

Параметр функции	Компонент типа ACH.1	Раздел, подраздел, пункт настоящего стандарта
GUIEventSubscriptionUuid	guiEventSubscriptionUuid	Раздел 19 совместно с 15.58
BSPUuid	bspUuid	Раздел 19 совместно с 15.58
BSPHandle	bspHandle	Раздел 19 совместно с 15.42
GUISelectEventHandler	guiSelectEventHandlerAddress	15.1.7
GUISelectEventHandler	guiSelectEventHandlerAddress	15.1.7
GUIStateEventHandler	guiStateEventHandlerAddress	15.1.7
GUIStateEventHandler	guiStateEventHandlerAddress	15.1.7
GUIProgressEventHandler	guiProgressEventHandlerAddress	15.1.7
GUIProgressEventHandlerCtx	guiProgressEventHandlerContext	15.1.7

16.24 Функция BioAPI_QueryGUIEventSubscriptions

16.24.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_QueryGUIEventSubscriptions
        (const BioAPI_UUID *BSPUuid,
        BioAPI_GUI_EVENT_SUBSCRIPTION **GUIEventSubscriptionList,
        uint32_t *NumberOfElements);
```

16.24.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **queryGUIEventSubscriptions** и тип сообщения ответа **queryGUIEventSubscriptions**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
QueryGUIEventSubscriptions-RequestParams ::= SEQUENCE {
        bspProductUuid BioAPI-UUID
    }
```

и

```
QueryGUIEventSubscriptions-ResponseParams ::= SEQUENCE {
        guiEventSubscriptions SEQUENCE (SIZE(0..max-unsigned-int)) OF
        subscription BioAPI-GUI-EVENT-SUBSCRIPTION
    }
```

16.24.3 Когда инфраструктура получает вызов к функции **BioAPI_QueryGUIEventSubscriptions** от локального приложения, она должна сначала определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23, и только затем выполняют действия, указанные в одном из следующих подпунктов.

16.24.3.1 Если главной конечной точкой является локальная конечная точка, инфраструктура выполняет следующие действия в указанном порядке:

a) создать временное абстрактное значение (*outgoingResponseParams*) типа **QueryGUIEventSubscriptions-ResponseParams** (см. 16.24.2), в котором компонент **guiEventSubscriptions** должен быть изначально пустым;

b) проверить таблицу **GUIEventLocalSubscriptions** (см. 18.10) на наличие поля, в котором:

1) присутствует необязательный компонент **guiEventSubscriptionUuid**;

2) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки и

3) компонент **bspProductUuid** имеет значение *bspProductUuid*;

c) для каждого такого поля (*localSubscription*) добавить элемент в компонент **guiEventSubscriptions** *outgoingResponseParams*, в котором:

1) компонент **subscriberEndpointIRI** должен быть установлен на ИИР локальной конечной точки;

2) компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** *localSubscription*;

3) в случае, если компонент **guiSelectEventHandlerAddress** *localSubscription* имеет значение, отличающееся от 0, компонент **guiSelectEventSubscribed** должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

4) в случае, если компонент **guiStateEventHandlerAddress** *localSubscription* имеет значение, отличающееся от 0, компонент **guiStateEventSubscribed** должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**; и

5) Если компонент **guiProgressEventHandlerAddress** *localSubscription* имеет значение, отличающееся от 0, компонент **guiProgressEventSubscribed** должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

d) проверить таблицу **GUIEventRemoteSubscriptions** (см. 18.11) на наличие всех полей, в которых присутствует необязательный компонент **guiEventSubscriptionUuid**, а компонент **bspProductUUid** имеет значение *bspProductUuid*;

e) для каждого такого значения (*remoteSubscription*) добавить элемент в компонент **guiEventSubscriptions** *outgoingResponseParams*, в котором

все компоненты установлены из компонентов *remoteSubscription* с такими же именами;

f) установить исходящие значения параметра вызова функции **BioAPI_QueryGUIEventSubscriptions** путем преобразования из *outgoingResponseParams* согласно 16.24.6;

g) возвратить значение 0 локальному приложению.

16.24.3.2 Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запрос/ответ ПМО БиоАПИ **queryGUIEventSubscriptions** согласно разделу 27, выполняя действия, указанные в 16.24.5 и 16.24.6 для преобразований между параметрами функции и компонентами ACH.1, если это установлено в указанном разделе.

16.24.3.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.24.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **queryGUIEventSubscriptions** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **QueryGUIEventSubscriptions-RequestParams** (см. 16.24.2) сообщения запроса ПМО БиоАПИ **queryGUIEventSubscriptions**;

b) создать временное абстрактное значение (*outgoingResponseParams*) типа **QueryGUIEventSubscriptions-ResponseParams** (см. 16.24.2), в котором компонент **guiEventSubscriptions** должен быть изначально пустым;

c) проверить таблицу **GUIEventLocalSubscriptions** (см. 18.10) на наличие всех полей, в которых:

1) присутствует необязательный компонент **guiEventSubscriptionUuid**;

2) компонент **hostingEndpointIRI** содержит ИИР второстепенной конечной точки и

3) компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** *incomingRequestParams*;

d) для каждого такого поля (*localSubscription*) добавить элемент в компонент **guiEventSubscriptions** *outgoingResponseParams*, в котором:

1) компонент **subscriberEndpointIRI** должен быть установлен на ИИР локальной конечной точки;

2) компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** *localSubscription*;

3) в случае, если компонент **guiSelectEventHandlerAddress** *localSubscription* имеет значение, отличающееся от 0, компонент **guiSelectEventSubscribed** должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

4) в случае, если компонент **guiStateEventHandlerAddress** *localSubscription* имеет значение, отличающееся от 0, компонент **guiStateEventSubscribed** должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**; и

5) в случае, если компонент **guiProgressEventHandlerAddress** *localSubscription* имеет значение, отличающееся от 0, компонент **guiProgressEventSubscribed** должен быть установлен на **TRUE**; в противном случае он должен быть установлен на **FALSE**;

e) проверить таблицу **GUIEventRemoteSubscriptions** (см. 18.11) на наличие всех полей, в которых присутствует необязательный компонент **guiEventSubscriptionUuid**, а компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** *incomingRequestParams*;

f) для каждого такого значения (*remoteSubscription*) добавить элемент в компонент **guiEventSubscriptions** *outgoingResponseParams*, в котором все компоненты будут установлены из компонентов *remoteSubscription* с такими же именами;

g) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **queryGUIEventSubscriptions** (см. 13.3) со значением параметра, установленным на *outgoingResponseParams*, и возвращаемым значением, установленным на 0.

16.24.5 Преобразование между параметрами функции Си **BioAPI_QueryGUIEventSubscriptions** и типом АСН.1 **QueryGUIEventSubscriptions-RequestParams** (см. 16.24.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 64.

Таблица 64 – Преобразование данных между параметрами функции **BioAPI_QueryGUIEventSubscriptions** и типом АСН.1 **QueryGUIEventSubscriptions-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел настоящего стандарта
BSPUuid	bspProductUuid	Раздел 25
GUIEventSubscriptionList	Отсутствует	Раздел 22
NumberOfElements	Отсутствует	Раздел 22

16.24.6 Преобразование между параметрами функции Си **BioAPI_QueryGUIEventSubscriptions** и типом АСН.1 **QueryGUIEventSubscriptions-ResponseParams** (см. 16.24.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 65.

Таблица 65 – Преобразование данных между параметрами функции **BioAPI_QueryGUIEventSubscriptions** и типом АСН.1 **QueryGUIEventSubscriptions-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
GUIEventSubscriptionList, NumberOfElements	guiEventSubscriptions	Раздел 20 совместно с 16.24.7 и 16.24.8

16.24.7 Преобразование двух переменных СИ, выделенных параметрами **GUIEventSubscriptionList/NumberOfElements**, в компонент АСН.1 **guiEventSubscriptions** выполняют следующим образом: принимают *N* равным значению переменной Си, выделенной параметром **NumberOfElements**; в этом случае первые элементы *N* типа **BioAPI_GUI_EVENT_SUBSCRIPTION** (см. 15.36) в массиве, выделенном переменной Си, которая выделена параметром **GUIEventSubscriptionList**, должны быть преобразованы по порядку в элемент компонента **guiEventSubscriptions** согласно 15.36. Компонент **guiEventSubscriptions** должен иметь точное число *N* элементов.

16.24.8 Преобразование компонента АСН.1 **guiEventSubscriptions** в две переменные Си, выделенных параметрами **GUIEventSubscriptionList/NumberOfElements**, выполняют следующим образом: принимаем *N* равным числу элементов компонента **guiEventSubscriptions**; в этом случае новый массив *N* элементов типа **BioAPI_GUI_EVENT_SUBSCRIPTION** (см. 15.36) должен быть заполнен путем преобразования каждого элемента компонента **guiEventSubscriptions** по порядку в элемент массива согласно 15.36. Переменная Си, выделенная параметром **GUIEventSubscriptionList**, должна быть установлена в адрес

массива, а переменная Си, выделенная параметром **NumberOfElements**, должна быть установлена в *N*.

16.25 Функция BioAPI_NotifyGUISelectEvent

16.25.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_NotifyGUISelectEvent
        (const uint8_t *SubscriberEndpointIRI,
        const BioAPI_UUID *GUIEventSubscriptionUuid,
        const BioAPI_UUID *BSPUuid,
        BioAPI_UNIT_ID UnitID,
        BioAPI_GUI_ENROLL_TYPE EnrollType,
        BioAPI_GUI_OPERATION Operation,
        BioAPI_GUI_MOMENT Moment,
        BioAPI_RETURN ResultCode,
        uint32_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);
```

16.25.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **notifyGUISelectEvent** и тип сообщения ответа **notifyGUISelectEvent**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
NotifyGUISelectEvent-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI           EndpointIRI,
        guiEventSubscriptionUuid        BioAPI-UUID,
        bspProductUuid                  BioAPI-UUID,
        unitID                          BioAPI-UNIT-ID,
        enrollType                      BioAPI-GUI-ENROLL-TYPE,
        operation                       BioAPI-GUI-OPERATION,
        moment                          BioAPI-GUI-MOMENT,
        resultCode                      BioAPI-RETURN,
        maxNumEnrollSamples             UnsignedInt,
        selectableInstances             BioAPI-BIR-SUBTYPE-MASK,
        capturedInstances               BioAPI-BIR-SUBTYPE-MASK,
        text UTF8String                 OPTIONAL
```

```
}
```

и

```
NotifyGUISelectEvent-ResponseParams ::= SEQUENCE {
    selectedInstances       BioAPI-BIR-SUBTYPE-MASK,
    response                BioAPI-GUI-RESPONSE
}
```

16.25.3 Когда инфраструктура получает вызов к функции **BioAPI_NotifyGUISelectEvent** от локального приложения, она должна сначала определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.25.3.1 Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*incomingRequestParams*) типа **NotifyGUISelectEvent-RequestParams** (см. 16.25.2) путем преобразования параметров вызова функции **BioAPI_NotifyGUISelectEvent** согласно 16.25.5;

b) создать временное абстрактное значение (*eventInfo*) типа **GUISelectEventInfo** (см. 16.2.4), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной конечной точки;

2) необязательный компонент **originalBSPHandle** должен отсутствовать и

3) оставшиеся компоненты должны быть установлены из *incomingRequestParams* с такими же именами;

c) зарегистрировать операцию выбора ГИП, основанную на *eventInfo*, в подписчике (либо обработчике операции выбора ГИП локального приложения или главной конечной точки) и определить значение параметра подтверждения (*incomingAcknowledgementParams*) и

возвращаемое значение параметра подтверждения (*incomingReturnValue*) согласно разделу 30;

d) Если *incomingReturnValue* не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

e) создать временное абстрактное значение (*outgoingResponseParams*) типа **NotifyGUISelectEvent-ResponseParams** (см. 16.25.2), в котором все компоненты должны быть установлены из компонентов *incomingAcknowledgementParams* с такими же именами;

f) установить исходящие параметры вызова функции **BioAPI_NotifyGUISelectEvent** путем преобразования из *outgoingResponseParams* согласно 16.25.6 и

g) возвратить значение 0 локальному приложению.

16.25.3.2 Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запрос/ответ ПМО БиоАПИ **notifyGUISelectEvent** согласно разделу 27, выполняя действия, указанные в 16.25.5 и 16.25.6 для преобразований между параметрами функции и компонентами ACH.1, если это установлено в указанном разделе.

16.25.3.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.25.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **notifyGUISelectEvent** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **NotifyGUISelectEvent- RequestParams** (см. 16.25.2) сообщения запроса ПМО БиоАПИ **notifyGUISelectEvent**;

b) создать временное абстрактное значение (*eventInfo*) типа **GUISelectEventInfo** (см. 16.2.4), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной конечной точки;

2) необязательный компонент **originalBSPHandle** должен отсутствовать и

3) оставшиеся компоненты должны быть установлены из *incomingRequestParams* с такими же именами;

c) зарегистрировать операцию выбора ГИП, основанную на *eventInfo*, в подписчике (либо обработчике операции выбора ГИП локального приложения или главной конечной точки) и определить значение параметра подтверждения (*incomingAcknowledgementParams*) и возвращаемое значение параметра подтверждения (*incomingReturnValue*) согласно разделу 30;

d) Если *incomingReturnValue* не равен 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **notifyGUISelectEvent** (см. 13.3) с возвращаемым значением, установленным на такое значение без выполнения следующих действий;

e) создать временное абстрактное значение (*outgoingResponseParams*) типа **NotifyGUISelectEvent-ResponseParams** (см. 16.25.2), в котором все компоненты должны быть установлены из компонентов *incomingAcknowledgementParams* с такими же именами;

f) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **notifyGUISelectEvent** (см. 13.3) со значением параметра, установленным на *outgoingResponseParams*, и возвращаемым значением, установленным на 0.

16.25.5 Преобразование между параметрами функции Си **BioAPI_NotifyGUISelectEvent** и типом АСН.1 **NotifyGUISelectEvent-RequestParams** (см. 16.25.2) выполняют путем преобразования между

индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 66.

Таблица 66 – Преобразование данных между параметрами функции **BioAPI_NotifyGUISelectEvent** и типом ACH.1 **NotifyGUISelectEvent-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
SubscriberEndpointIRI	subscriberEndpointIRI	15.3
GUIEventSubscriptionUuid	guiEventSubscriptionUuid	Раздел 19 совместно с 15.58
BSPUuid	bspProductUuid	Раздел 25
UnitID	unitID	15.55
EnrollType	enrollType	15.38
Operation	operation	15.39
Moment	moment	15.37
ResultCode	resultCode	15.52
MaxNumEnrollSamples	maxNumEnrollSamples	15.1.6
SelectableInstances	selectableInstances	15.17
CapturedInstances	capturedInstances	15.17
Text	text	15.2
SelectedInstances	Отсутствует	Раздел 22
Response	Отсутствует	Раздел 22

16.25.6 Преобразование между параметрами функции Си **BioAPI_NotifyGUISelectEvent** и типом ACH.1 **NotifyGUISelectEvent-ResponseParams** (см. 16.25.2) выполняют путем преобразования между

индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 67.

Таблица 67 – Преобразование данных между параметрами функции **BioAPI_NotifyGUISelectEvent** и типом ACH.1 **NotifyGUISelectEvent-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, подраздел настоящего стандарта
SelectedInstances	selectedInstances	Раздел 20 совместно с 15.17
Response	response	Раздел 20 совместно с 15.40

16.26 Функция BioAPI_NotifyGUIStateEvent

16.26.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_NotifyGUIStateEvent
    (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,
    BioAPI_RETURN ResultCode,
    int32_t EnrollSampleIndex,
    const BioAPI_GUI_BITMAP_ARRAY *Bitmaps,
    const uint8_t *Text,
    BioAPI_GUI_RESPONSE *Response,
    int32_t *EnrollSampleIndexToRecapture);
```

16.26.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **notifyGUIStateEvent** и тип сообщения ответа **notifyGUIStateEvent**, которые переносят значение

следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
NotifyGUIStateEvent-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI EndpointIRI,
        guiEventSubscriptionUuid BioAPI-UUID,
        bspProductUuid BioAPI-UUID,
        unitID BioAPI-UNIT-ID,
        operation BioAPI-GUI-OPERATION,
        suboperation BioAPI-GUI-SUBOPERATION,
        purpose BioAPI-BIR-PURPOSE,
        moment BioAPI-GUI-MOMENT,
        resultCode BioAPI-RETURN,
        enrollSampleIndex SignedInt,
        bitmaps BioAPI-GUI-BITMAP-ARRAY OPTIONAL,
        text UTF8String OPTIONAL
    }
```

и

```
NotifyGUIStateEvent-ResponseParams ::= SEQUENCE {
        response BioAPI-GUI-RESPONSE,
        enrollSampleIndexToRecapture SignedInt
    }
```

16.26.3 Когда инфраструктура получает вызов к функции **BioAPI_NotifyGUIStateEvent** от локального приложения, она должна сначала определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.26.3.1 Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*incomingRequestParams*) типа **NotifyGUIStateEvent- RequestParams** (см. 16.26.2) путем преобразования параметров вызова функции **BioAPI_NotifyGUIStateEvent** согласно 16.26.5;

b) создать временное абстрактное значение (*eventInfo*) типа **GUIStateEventInfo** (см. 17.3.4), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной конечной точки;

2) необязательный компонент **originalBSPHandle** должен отсутствовать и

3) оставшиеся компоненты должны быть установлены из *incomingRequestParams* с такими же именами;

c) зарегистрировать операцию выбора ГИП, основанную на *eventInfo,* в подписчике (либо обработчике операции выбора ГИП локального приложения или главной конечной точки) и определить значение параметра подтверждения (*incomingAcknowledgementParams*) и возвращаемое значение параметра подтверждения (*incomingReturnValue*) согласно разделу 31;

d) Если *incomingReturnValue* не равен 0, возвратить такое значение локальному приложению без выполнения следующих действий;

e) создать временное абстрактное значение (читай *outgoingResponseParams*) типа **NotifyGUIStateEvent-ResponseParams** (см. 16.26.2), в котором все компоненты должны быть установлены из компонентов *incomingAcknowledgementParams* с такими же именами;

f) установить исходящие параметры вызова функции **BioAPI_NotifyGUIStateEvent** путем преобразования из *outgoingResponseParams* согласно 16.26.6 и

g) возвратить значение 0 локальному приложению.

16.26.3.2 Если главная конечная точка это второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запрос/ответ ПМО БиоАПИ **notifyGUIStateEvent** согласно разделу 27, выполняя действия, указанные в 16.26.5 и 16.26.6 для преобразований между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.26.3.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.26.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **notifyGUIStateEvent notifyGUISelectEvent** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **NotifyGUIStateEvent-RequestParams** (см. 16.26.2) сообщения запроса ПМО БиоАПИ **notifyGUIStateEvent**;

b) создать временное абстрактное значение (*eventInfo*) типа **GUIStateEventInfo** (см. 17.3.4), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной конечной точки;

2) необязательный компонент **originalBSPHandle** должен отсутствовать и

3) оставшиеся компоненты должны быть установлены из *incomingRequestParams* с такими же именами;

c) зарегистрировать ГИП состояния операции, основанную на *eventInfo*, на подписчика (либо обработчике ГИП состояния операции локального приложения или главной конечной точки) и определить значение параметра подтверждения (*incomingAcknowledgementParams*) и возвращаемое значение параметра подтверждения (*incomingReturnValue*) согласно разделу 31;

d) Если *incomingReturnValue* не равен 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **notifyGUIStateEvent** (см. 13.3) с возвращаемым значением, установленным на такое значение без выполнения следующих действий;

e) создать временное абстрактное значение (*outgoingResponseParams*) типа **NotifyGUIStateEvent- ResponseParams** (см. 16.26.2), в котором все компоненты должны быть установлены из компонентов *incomingAcknowledgementParams* с такими же именами;

f) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **notifyGUIStateEvent** (см. 13.3) со значением параметра, установленным на *outgoingResponseParams*, и возвращаемым значением, установленным на 0.

16.26.5 Преобразование между параметрами функции Си **BioAPI_NotifyGUIStateEvent** и типом АСН.1 **NotifyGUIStateEvent-RequestParams** (см. 16.26.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 68.

Таблица 68 – Преобразование данных между параметрами функции **BioAPI_NotifyGUIStateEvent** и типом АСН.1 **NotifyGUIStateEvent-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
SubscriberEndpointIRI	subscriberEndpointIRI	15.3
GUIEventSubscriptionUuid	guiEventSubscriptionUuid	Раздел 19 совместно с 15.58
BSPUuid	bspProductUuid	Раздел 25
UnitID	unitID	15.55
Operation	operation	15.39
Suboperation	suboperation	15.41
Purpose	purpose	15.14

Окончание таблицы 68

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
Moment	moment	15.37
ResultCode	resultCode	15.52
EnrollSampleIndex	enrollSampleIndex	15.1.6
Bitmaps	bitmaps	Раздел 19 совместно с 15.35
Text	text	15.2
Response	*отсутствует*	Раздел 22
EnrollSampleIndexToRecapture	*отсутствует*	Раздел 22

16.26.6 Преобразование между параметрами функции Си **BioAPI_NotifyGUIStateEvent** и типом ACH.1 **NotifyGUIStateEvent-ResponseParams** (см. 16.26.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 69.

Таблица 69 – Преобразование данных между параметрами функции **BioAPI_NotifyGUIStateEvent** и типом ACH.1 **NotifyGUIStateEvent-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
EnrollSampleIndexToRecapture	enrollSampleIndexToRecapture	Раздел 20 совместно с 15.1.6
Response	response	Раздел 20 совместно с 15.40

16.27 Функция BioAPI_NotifyGUIProgressEvent

16.27.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.27.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **notifyGUIProgressEvent** и тип сообщения ответа **notifyGUIProgressEvent**, которые переносят значение

следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
NotifyGUIProgressEvent-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI EndpointIRI,
        guiEventSubscriptionUuid BioAPI-UUID,
        bspProductUuid BioAPI-UUID,
        unitID BioAPI-UNIT-ID,
        operation BioAPI-GUI-OPERATION,
        suboperation BioAPI-GUI-SUBOPERATION,
        purpose BioAPI-BIR-PURPOSE,
        moment BioAPI-GUI-MOMENT,
        suboperationProgress UnsignedByte,
        bitmaps BioAPI-GUI-BITMAP-ARRAY OPTIONAL,
        text UTF8String OPTIONAL
    }
```

и

```
NotifyGUIProgressEvent-ResponseParams ::= SEQUENCE {
        response BioAPI-GUI-RESPONSE
    }
```

16.27.3 Когда инфраструктура получает вызов к функции **BioAPI_NotifyGUIProgressEvent** от локального приложения, она должна сначала определить главную конечную точку и УУИД продукта ПБУ (*bspProductUuid*) из параметра **BSPUuid** согласно разделу 23, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.27.3.1 Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*incomingRequestParams*) типа **NotifyGUIProgressEvent-RequestParams** (см. 16.27.2) путем преобразования параметров вызова функции **BioAPI_NotifyGUIProgressEvent** согласно 16.27.5;

b) создать временное абстрактное значение (*eventInfo*) типа **GUIProgressEventInfo** (см. 17.4.4) , в котором:

1) компонент **hostingEndpointIRI** должен быть установлен в ИИР локальной конечной точки;

2) необязательный компонент **originalBSPHandle** должен отсутствовать и

3) оставшиеся компоненты должны быть установлены из *incomingRequestParams* с такими же именами;

c) зарегистрировать операцию прогресса ГИП, основанную на *eventInfo,* в подписчике (либо обработчике операции прогресса ГИП локального приложения или главной конечной точки) и определить значение параметра подтверждения (*incomingAcknowledgementParams*) и возвращаемое значение параметра подтверждения (*incomingReturnValue*) согласно разделу 32;

d) Если *incomingReturnValue* не равен 0, возвратить это значение локальному приложению без выполнения следующих действий;

e) создать временное абстрактное значение (*outgoingResponseParams*) типа **NotifyGUIProgressEvent- ResponseParams** (см. 16.27.2), в котором все компоненты должны быть установлены из компонентов *incomingAcknowledgementParams* с такими же именами;

f) установить исходящие параметры вызова функции **BioAPI_NotifyGUIProgressEvent** путем преобразования *outgoingResponseParams* согласно 16.27.6; и

g) возвратить значение 0 локальному приложению.

16.27.3.2 Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запрос/ответ ПМО БиоАПИ **notifyGUIProgressEvent** согласно разделу 27, выполняя действия, указанные в 16.27.5 и 16.27.6, для преобразований между параметрами функции и компонентами АСН.1, если это установлено в данном разделе.

16.27.3.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.27.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **notifyGUIProgressEvent** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **NotifyGUIProgressEvent- RequestParams** (см. 16.27.2) сообщения запроса ПМО БиоАПИ **notifyGUIProgressEvent**;

b) создать временное абстрактное значение (*eventInfo*) типа **GUIProgressEventInfo** (см. 17.4.4), в котором:

1) компонент **hostingEndpointIRI** должен быть установлен на ИИР локальной конечной точки;

2) необязательный компонент **originalBSPHandle** должен отсутствовать и

3) оставшиеся компоненты должны быть установлены из *incomingRequestParams* с такими же именами;

c) зарегистрировать операцию прогресса ГИП, основанную на *eventInfo,* в подписчике (либо обработчике операции прогресса ГИП локального приложения или главной конечной точки) и определить значение параметра подтверждения (*incomingAcknowledgementParams*) и возвращаемое значение параметра подтверждения (*incomingReturnValue*) согласно разделу 32;

d) в случае, если *incomingReturnValue* не равен 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **notifyGUIProgressEvent** (см. 13.3) с возвращаемым значением, установленным на это значение без выполнения следующих действий;

e) создать временное абстрактное значение (*outgoingResponseParams*) типа **NotifyGUIProgressEvent- ResponseParams** (см. 16.27.2), в

котором все компоненты должны быть установлены из компонентов *incomingAcknowledgementParams* с такими же именами;

f) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **notifyGUIProgressEvent** (см. 13.3) со значением параметра, установленным на *outgoingResponseParams*, и возвращаемым значением, установленным на 0.

16.27.5 Преобразование между параметрами функции Си **BioAPI_NotifyGUIProgressEvent** и типом ACH.1 **NotifyGUIProgressEvent-RequestParams** (см. 16.27.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 70.

Таблица 70 – Преобразование данных между параметрами функции **BioAPI_NotifyGUIProgressEvent** и типом ACH.1 **NotifyGUIProgressEvent-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
SubscriberEndpointIRI	subscriberEndpointIRI	15.3
GUIEventSubscriptionUuid	guiEventSubscriptionUuid	Раздел 19 совместно с 15.58
BSPUuid	bspProductUuid	Раздел 25
UnitID	unitID	15.55
Operation	operation	15.39
Suboperation	suboperation	15.41
Purpose	purpose	15.14
Moment	moment	15.37
SuboperationInProgress	suboperationProgress	15.1.3
Bitmaps	Bitmaps	Раздел 19 совместно с 15.35
Text	Text	15.2
Response	Отсутствует	Раздел 22

16.27.6 Преобразование между параметрами функции Си **BioAPI_NotifyGUIProgressEvent** и типом АСН.1 **NotifyGUIProgressEvent-ResponseParams** (см. 16.27.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 71.

Таблица 71 – Преобразование данных между параметрами функции **BioAPI_NotifyGUIProgressEvent** и типом АСН.1 **NotifyGUIProgressEvent-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
Response	response	Раздел 20 совместно с 15.40

16.28 Функция BioAPI_RedirectGUIEvents

16.28.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.28.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **redirectGUIEvents** и тип сообщения ответа **redirectGUIEvents**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
RedirectGUIEvents-RequestParams ::= SEQUENCE {
                subscriberEndpointIRI EndpointIRI,
                guiEventSubscriptionUuid BioAPI-UUID,
                originalBSPHandle BioAPI-HANDLE,
                guiSelectEventRedirected BOOLEAN,
                guiStateEventRedirected BOOLEAN,
```

guiProgressEventRedirected BOOLEAN

}

и

RedirectGUIEvents-ResponseParams ::= NULL

16.28.3 Когда инфраструктура получает вызов к функции **BioAPI_RedirectGUIEvents** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.28.3.1 Если главной конечной точкй является локальная конечная точка, инфраструктура выполняет следующие действия в указанном порядке:

a) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с такими же значениями параметра, как и во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования из *originalBSPHandle* согласно 15.42;

b) Если возвращенное значение внутреннего вызова не является 0, возвратить это значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*incomingRequestParams*) типа **RedirectGUIEvents-RequestParams** (см. 16.28.2) путем преобразования параметров вызова функции **BioAPI_RedirectGUIEvents** согласно 16.28.5;

d) добавить поле в таблицу **GUIEventRedirectors** (см. 18.12), в котором:

1) компоненты **referrerEndpointIRI** и **bspProductUuid** должны быть установлены из компонентов с такими же именами из поля таблицы **AttachSessionRemoteReferences** (см. 18.9), в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** *incomingRequestParams* и

2) оставшиеся компоненты должны быть установлены из компонентов *incomingRequestParams* с такими же именами;

e) возвратить значение 0 локальному приложению.

16.28.3.2 Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запрос/ответ ПМО БиоАПИ **redirectGUIEvents** согласно разделу 27, выполняя действия, указанные в 16.28.5 для преобразований между параметрами функции и компонентами ACH.1, если это установлено в указанном разделе.

16.28.3.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.28.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **redirectGUIEvents** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **RedirectGUIEvents-RequestParams** (см. 16.28.2) сообщения запроса ПМО БиоАПИ **redirectGUIEvents**;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_RedirectGUIEvents**, в котором параметры вызова функции должны быть установлены путем преобразования из *incomingRequestParams* согласно 16.28.5;

c) Если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **redirectGUIEvents** (см. 13.3) с возвращаемым значением установленным на это значение без выполнения следующих действий;

d) добавить поле в таблицу **GUIEventRedirectors** (см. 18.12), в котором:

1) компоненты **referrerEndpointIRI** и **bspProductUuid** должны быть установлены из компонентов с такими же именами из поля таблицы **AttachSessionRemoteReferences** (см. 18.9), в котором компонент

originalBSPHandle имеет такое значение, как и компонент **originalBSPHandle** *incomingRequestParams* и

2) оставшиеся компоненты должны быть установлены из компонентов *incomingRequestParams* с такими же именами;

e) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **redirectGUIEvents** (см. 13.3) со значением параметра, установленным на **NULL**, и с возвращаемым значением, установленным на 0.

16.28.5 Преобразование между параметрами функции Си **BioAPI_RedirectGUIEvents** и типом ACH.1 **RedirectGUIEvents-RequestParams** (см. 16.28.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 72.

Таблица 72 – Преобразование данных между параметрами функции **BioAPI_RedirectGUIEvents** и типом ACH.1 **RedirectGUIEvents-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
SubscriberEndpointIRI	subscriberEndpointIRI	15.3
GUIEventSubscriptionUuid	guiEventSubscriptionUuid	Раздел 19 совместно с 15.58
BSPHandle	originalBSPHandle	Раздел 26
GUISelectEventRedirected	guiSelectEventRedirected	15.18
GUIStateEventRedirected	guiStateEventRedirected	15.18
GUIProgressEventRedirected	guiProgressEventRedirected	15.18

16.29 Функция BioAPI_UnredirectGUIEvents

16.29.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.29.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **unredirectGUIEvents** и тип сообщения ответа **unredirectGUIEvents**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
UnredirectGUIEvents-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI           EndpointIRI,
        guiEventSubscriptionUuid        BioAPI-UUID,
        originalBSPHandle               BioAPI-HANDLE,
        guiSelectEventRedirected        BOOLEAN,
        guiStateEventRedirected         BOOLEAN,
        guiProgressEventRedirected      BOOLEAN
    }
```

и

```
UnredirectGUIEvents-ResponseParams ::= NULL
```

16.29.3 Когда инфраструктура получает вызов к функции **BioAPI_UnredirectGUIEvents** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.29.3.1 Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить следующие действия в указанном порядке:

а) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с такими же значениями параметра, как и во входящем вызове,

за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования из *originalBSPHandle* согласно 15.42;

b) Если возвращенное значение внутреннего вызова не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*incomingRequestParams*) типа **UnredirectGUIEvents-RequestParams** (см. 16.29.2) путем преобразования параметров вызова функции **BioAPI_UnredirectGUIEvents** согласно 16.29.5;

d) проверить таблицу **GUIEventRedirectors** (см. 18.12 на наличие поля, в котором компоненты **subscriberEndpointIRI**, **guiEventSubscriptionUuid**, **originalBSPHandle**, **guiSelectEventRedirected**, **guiStateEventRedirected** и **guiProgressEventRedirected** имеют такие же значения, как и компоненты *incomingRequestParams* с такими же именами;

e) Если такое поле не обнаружено, возвратить значение **BioAPIERR_NO_SUCH_REDIRECTOR_FOUND** локальному приложению без выполнения следующих действий;

f) удалить поле таблицы **GUIEventRedirectors** (выполняют действия, указанные в 18.12.3).

Примечание – Если обнаружено несколько полей, любое из них (только одно) должно быть удалено;

g) возвратить значение 0 локальному приложению.

16.29.3.2 Если главная конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запрос/ответ ПМО БиоАПИ **unredirectGUIEvents** как указано в разделе 27, выполняя действия, указанные в 16.29.5 для преобразований между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.29.3.3 Если главная конечная точка не может быть определена, инфраструктура должна вернуть значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.29.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **unredirectGUIEvents** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **UnredirectGUIEvents-RequestParams** (см. 16.29.2) сообщения запроса ПМО БиоАПИ **unredirectGUIEvents**;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnredirectGUIEvents**, в котором параметры вызова функции должны быть установлены путем преобразования из *incomingRequestParams* согласно 16.29.5;

c) Если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unredirectGUIEvents** (см. 13.3) с возвращаемым значением, установленным на это значение без выполнения следующих действий;

d) проверить таблицу **GUIEventRedirectors** (см. 18.12) на наличие поля, в котором компоненты **subscriberEndpointIRI**, **guiEventSubscriptionUuid**, **originalBSPHandle**, **guiSelectEventRedirected**, **guiStateEventRedirected** и **guiProgressEventRedirected** имеют такие же значения, как и компоненты *incomingRequestParams* с такими же именами;

e) в случае, если такое поле не обнаружено, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unredirectGUIEvents** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_NO_SUCH_REDIRECTOR_FOUND** без выполнения следующих действий;

f) удалить поле таблицы **GUIEventRedirectors** (выполняют действия, указанные в 18.12.3).

Примечание – Если обнаружено несколько полей, любое из них (только одно) будет удалено;

g) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unredirectGUIEvents** (см. 13.3) со значением параметра, установленным на **NULL**, и с возвращаемым значением, установленным на 0.

17.29.5 Преобразование между параметрами функции Си **BioAPI_UnredirectGUIEvents** и типом ACH.1 **UnredirectGUIEvents-RequestParams** (см. 16.29.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 73.

Таблица 73 – Преобразование данных между параметрами функции **BioAPI_UnredirectGUIEvents** и типом ACH.1 **UnredirectGUIEvents-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
SubscriberEndpointIRI	subscriberEndpointIRI	15.3
GUIEventSubscriptionUuid	guiEventSubscriptionUuid	Раздел 19 совместно с 15.58
BSPHandle	originalBSPHandle	Раздел 26
GUISelectEventRedirected	guiSelectEventRedirected	15.18
GUIStateEventRedirected	guiStateEventRedirected	15.18
GUIProgressEventRedirected	guiProgressEventRedirected	15.18

16.30 Функция BioAPI_Capture

16.30.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.30.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **capture** и тип сообщения ответа **capture**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
Capture-RequestParams ::= SEQUENCE {
        originalBSPHandle    BioAPI-HANDLE,
        purpose              BioAPI-BIR-PURPOSE,
        subtype              BioAPI-BIR-SUBTYPE,
        outputFormat         BioAPI-BIR-BIOMETRIC-DATA-FORMAT OPTIONAL,
        timeout              SignedInt,
        no-auditData         BOOLEAN
    }
```

и

```
Capture-ResponseParams ::= SEQUENCE {
        capturedBIR          BioAPI-BIR-HANDLE,
        auditData            BioAPI-BIR-HANDLE OPTIONAL
    }
```

16.30.3 Когда инфраструктура получает вызов к функции **BioAPI_Capture** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точкая, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с такими же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования

originalBSPHandle согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **capture** согласно разделу 27, выполняя действия, указанные в 16.30.5 и 16.30.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.30.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **capture** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Capture** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **capture** согласно разделу 28, выполняя действия, указанные в 16.30.5 и 16.30.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.30.5 Преобразование между параметрами функции Си **BioAPI_Capture** и типом АСН.1 **Capture-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 74.

Таблица 74 – Преобразование данных между параметрами функции **BioAPI_Capture** и типом ACH.1 **Capture-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
Purpose	purpose	15.14
Subtype	subtype	15.16
OutputFormat	outputFormat	Раздел 19 в соответствии с 15.8
CapturedBIR	Отсутствует	Раздел 22
Timeout	timeout	15.1.6
AuditData	no-auditData	Раздел 21

16.30.6 Преобразование между параметрами функции Си **BioAPI_Capture** и типом ACH.1 **Capture-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 75.

Таблица 75 – Преобразование данных между параметрами функции **BioAPI_Capture** и типом ACH.1 **Capture-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
CapturedBIR	capturedBIR	Раздел 20 совместно с 15.12
AuditData	auditData	Раздел 20 совместно с 15.12

16.31 Функция BioAPI_CreateTemplate

16.31.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.31.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **createTemplate** и тип сообщения ответа **createTemplate**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
CreateTemplate-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        capturedBIR             BioAPI-INPUT-BIR,
        referenceTemplate       BioAPI-INPUT-BIR OPTIONAL,
        outputFormat            BioAPI-BIR-BIOMETRIC-DATA-FORMAT OPTIONAL,
        payload                 BioAPI-DATA OPTIONAL,
        no-templateUuid         BOOLEAN
    }
```

и

```
CreateTemplate-ResponseParams ::= SEQUENCE {
        newTemplate             BioAPI-BIR-HANDLE,
        templateUuid            BioAPI-UUID OPTIONAL
    }
```

16.31.3 Когда инфраструктура получает вызов к функции **BioAPI_CreateTemplate** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкй является локальная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с такими же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем

преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **createTemplate** согласно разделу 27, выполняя действия, указанные в 16.31.5 и 16.31.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.31.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **createTemplate** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_CreateTemplate** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **createTemplate** согласно разделу 28, выполняя действия, указанные в 16.31.5 и 16.31.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.31.5 Преобразование между параметрами функции Си **BioAPI_CreateTemplate** и типом АСН.1 **CreateTemplate-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 76.

Таблица 76 – Преобразование данных между параметрами функции **BioAPI_CreateTemplate** и типом ACH.1 **CreateTemplate-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
CapturedBIR	capturedBIR	Раздел 19 совместно с 15.46
ReferenceTemplate	referenceTemplate	Раздел 19 совместно с 15.46
OutputFormat	outputFormat	Раздел 19 совместно с 15.8
NewTemplate	Отсутствует	Раздел 22
Payload	payload	Раздел 19 совместно с 15.22
TemplateUuid	no-templateUuid	Раздел 21

16.31.6 Преобразование между параметрами функции Си **BioAPI_CreateTemplate** и типом ACH.1 **CreateTemplate-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 77.

Таблица 77 – Преобразование данных между параметрами функции **BioAPI_CreateTemplate** и типом ACH.1 **CreateTemplate-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
NewTemplate	newTemplate	Раздел 20 совместно с 15.12
TemplateUuid	templateUuid	Раздел 20 совместно с 15.58

16.32 Функция BioAPI_Process

16.32.1 Данная функция определена в БиоАПИ следующим образом:

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

16.32.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **process** и тип сообщения ответа **process**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
Process-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                capturedBIR          BioAPI-INPUT-BIR,
                outputFormat         BioAPI-BIR-BIOMETRIC-DATA-FORMAT OPTIONAL
        }
```

и

```
Process-ResponseParams ::= SEQUENCE {
                ProcessedBIR         BioAPI-BIR-HANDLE
        }
```

16.32.3 Когда инфраструктура получает вызов к функции **BioAPI_Process** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **process** согласно разделу 27,

выполняя действия, указанные в 16.32.5 и 16.32.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.32.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **process** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Process** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **process** согласно разделу 28, выполняя действия, указанные в 16.32.5 и 16.32.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.32.5 Преобразование между параметрами функции Си **BioAPI_Process** и типом АСН.1 **Process-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 78.

Таблица 78 – Преобразования данных между параметрами функции **BioAPI_Process** и типом АСН.1 **Process-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
CapturedBIR	capturedBIR	Раздел 19 совместно с 15.46
OutputFormat	outputFormat	Раздел 19 совместно с 15.18
ProcessedBIR	Отсутствует	Раздел 22

16.32.6 Преобразование между параметрами функции Си **BioAPI_Process** и типом АСН.1 **Process-ResponseParams** выполняют

путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 79.

Таблица 79 – Преобразование данных между параметрами функции **BioAPI_Process** и типом АСН.1 **Process-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
ProcessedBIR	processedBIR	Раздел 20 совместно с 15.12

16.33 Функция BioAPI_ProcessWithAuxBIR

16.33.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_Return BioAPI BioAPI_ProcessWithAuxBIR
        (BioAPI_HANDLE BSPHandle,
        const BioAPI_INPUT_BIR *CapturedBIR,
        const BioAPI_INPUT_BIR *AuxiliaryData,
        const BioAPI_BIR_BIOMETRIC_DATA_FORMAT *OutputFormat,
        BioAPI_HANDLE *ProcessedBIR);
```

16.33.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **processWithAuxBIR** и тип сообщения ответа **processWithAuxBIR**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
ProcessWithAuxBIR-RequestParams ::= SEQUENCE {
                originalBSPHandle     BioAPI-HANDLE,
                capturedBIR           BioAPI-INPUT-BIR,
                auxiliaryData         BioAPI-INPUT-BIR,
                outputFormat          BioAPI-BIR-BIOMETRIC-DATA-FORMAT OPTIONAL
        }
```

и

```
ProcessWithAuxBIR-ResponseParams ::= SEQUENCE {
                processedBIR          BioAPI-BIR-HANDLE
        }
```

16.33.3 Когда инфраструктура получает вызов к функции **BioAPI_ProcessWithAuxBIR** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечня точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **processWithAuxBIR** согласно разделу 27, выполняя действия, указанные в 16.33.5 и 16.33.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.33.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **processWithAuxBIR** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_ProcessWithAuxBIR** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **processWithAuxBIR** согласно разделу 28, выполняя действия, указанные в 16.33.5 и 16.33.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.33.5 Преобразование между параметрами функции Си **BioAPI_ProcessWithAuxBIR** и типом АСН.1 **ProcessWithAuxBIR-**

RequestParams выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 80.

Таблица 80 – Преобразования данных между параметрами функции **BioAPI_ProcessWithAuxBIR** и типом ACH.1 **ProcessWithAuxBIR-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
CapturedBIR	capturedBIR	Раздел 19 совместно с 15.46
AuxiliaryData	auxiliaryData	Раздел 19 совместно с 15.46
OutputFormat	outputFormat	Раздел 19 совместно с 15.8
ProcessedBIR	Отсутствует	Раздел 22

16.33.6 Преобразование между параметрами функции Си **BioAPI_ProcessWithAuxBIR** и типом ACH.1 **ProcessWithAuxBIR-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 81.

Таблица 81 – Преобразование данных между параметрами функции **BioAPI_ProcessWithAuxBIR** и типом ACH.1 **ProcessWithAuxBIR-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
ProcessedBIR	processedBIR	Раздел 20 совместно с 15.12

16.34 Функция BioAPI_VerifyMatch

16.34.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.34.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **VerifyMatch** и тип сообщения ответа **VerifyMatch**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
VerifyMatch-RequestParams ::= SEQUENCE {
                originalBSPHandle       BioAPI-HANDLE,
                maxFMRRequested         BioAPI-FMR,
                processedBIR            BioAPI-INPUT-BIR,
                referenceTemplate       BioAPI-INPUT-BIR,
                no-adaptedBIR           BOOLEAN,
                no-fmrAchieved          BOOLEAN,
                no-payload              BOOLEAN
        }
```

и

```
VerifyMatch-ResponseParams ::= SEQUENCE {
                adaptedBIR              BioAPI-BIR-HANDLE OPTIONAL,
                result                  BOOLEAN,
                fmrAchieved             BioAPI-FMR OPTIONAL,
                payload                 BioAPI-DATA OPTIONAL
        }
```

16.34.3 Когда инфраструктура получает вызов к функции **BioAPI_VerifyMatch** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечня точка, инфраструктура

должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **VerifyMatch** согласно разделу 27, выполняя действия, указанные в 16.34.5 и 16.34.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.34.4 Когда инфраструктура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **VerifyMatch** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_VerifyMatch** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **VerifyMatch** согласно разделу 27, выполняя действия, указанные в 16.34.5 и 16.34.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.34.5 Преобразование между параметрами функции Си **BioAPI_VerifyMatch** и типом АСН.1 **VerifyMatch-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 82.

Таблица 82 – Преобразование данных между параметрами функции **BioAPI_VerifyMatch** и типом ACH.1 **VerifyMatch-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
MaxFMRRequested	maxFMRRequested	15.32
ProcessedBIR	processedBIR	Раздел 19 совместно с 15.46
ReferenceTemplate	referenceTemplate	Раздел 19 совместно с 15.46
AdaptedBIR	no-adaptedBIR	Раздел 21
Result	Отсутствует	Раздел 22
FMRAchieved	no-fmrAchieved	Раздел 21
Payload	no-payload	Раздел 21

16.34.6 Преобразование между параметрами функции Си **BioAPI_VerifyMatch** и типом ACH.1 **VerifyMatch-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно Таблице 83.

Таблица 83 – Преобразования данных между параметрами функции **BioAPI_VerifyMatch** и типом ACH.1 **VerifyMatch-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
AdaptedBIR	adaptedBIR	Раздел 20 совместно с 15.12
Result	result	Раздел 20 совместно с 15.18
FMRAchieved	fmrAchieved	Раздел 20 совместно с 15.32
Payload	payload	Раздел 20 совместно с 15.22

16.35 Функция BioAPI_IdentifyMatch

16.35.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.35.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **identifyMatch** и тип сообщения ответа **identifyMatch**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
IdentifyMatch-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        maxFMRRequested         BioAPI-FMR,
        processedBIR            BioAPI-INPUT-BIR,
        population              BioAPI-IDENTIFY-POPULATION,
        totalNumberOfTemplates  UnsignedInt,
        binning                 BOOLEAN,
        maxNumberOfResults      UnsignedInt,
        timeout                 SignedInt
    }
```

и

```
IdentifyMatch-ResponseParams ::= SEQUENCE {
        Candidates              SEQUENCE  (SIZE(0..max-unsigned-int)) OF
        candidate                   BioAPI-CANDIDATE
    }
```

16.35.3 Когда инфраструктура получает вызов к функции **BioAPI_IdentifyMatch** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если

главной конечной точкой является локальная конечная точка, инфраструктура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка инфраструктуры, инфраструктура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **identifyMatch** согласно разделу 27, выполняя действия, указанные в 16.35.5 и 16.35.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, инфраструктура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.35.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **identifyMatch** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_IdentifyMatch** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **identifyMatch** согласно разделу 28, выполняя действия, указанные в 16.35.5 и 16.35.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.35.5 Преобразование между параметрами функции Си **BioAPI_IdentifyMatch** и типом АСН.1 **IdentifyMatch-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 84.

Таблица 84 – Преобразование данных между параметрами функции **BioAPI_IdentifyMatch** и типом ACH.1 **IdentifyMatch-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
MaxFMRRequested	maxFMRRequested	15.32
ProcessedBIR	processedBIR	Раздел 19 совместно с 15.46
Population	population	Раздел 19 совместно с 15.43
TotalNumberOfTemplates	totalNumberOfTemplates	15.1.5
Binning	binning	15.18
MaxNumberOfResults	maxNumberOfResults	15.1.5
NumberOfResults	Отсутствует	Раздел 22
Candidates	Отсутствует	Раздел 22
Timeout	timeout	15.1.6

16.35.6 Преобразование между параметрами функции Си **BioAPI_IdentifyMatch** и типом ACH.1 **IdentifyMatch-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 85.

Таблица 85 – Преобразование данных между параметрами функции **BioAPI_IdentifyMatch** и типом ACH.1 **IdentifyMatch-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
NumberOfResults, Candidates	candidates	Раздел 20 совместно с 16.35.7 и 16.35.8

16.35.7 Преобразование двух переменных СИ, выделенных параметрами **Candidates/NumberOfResults**, в компонент ACH.1 **candidates** выполняют следующим образом: принимают *N* равным значению переменной Си, выделенной параметром **NumberOfResults**; в этом случае первые элементы *N* (типа **BioAPI_CANDIDATE** – см. 14.20) в массиве, выделенном переменной Си, которая выделена параметром **Candidates**, должны быть преобразованы по порядку в элемент компонента **candidates** согласно 15.36. Компонент **candidates** должен иметь точное число *N* элементов.

16.35.8 Преобразование компонента ACH.1 **candidates** в две переменных Си, выделенных параметрами **Candidates/NumberOfResults**, выполняют следующим образом: принимают *N* равным числу элементов компонента **candidates**; в этом случае новый массив *N* элементов типа **BioAPI_CANDIDATE** (см. 15.20) должен быть заполнен путем преобразования каждого элемента компонента **candidates** по порядку в элемент массива согласно 15.20. Переменная Си, выделенная параметром **Candidates**, должна быть установлена в адрес массива, а переменная Си, выделенная параметром **NumberOfResults**, должна быть установлена в *N*.

16.36 Функция BioAPI_Enroll

16.36.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.36.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **enroll** и тип сообщения ответа **enroll**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
Enroll-RequestParams ::= SEQUENCE {
        originalBSPHandle    BioAPI-HANDLE,
        purpose              BioAPI-BIR-PURPOSE,
        subtype              BioAPI-BIR-SUBTYPE,
        outputFormat         BioAPI-BIR-BIOMETRIC-DATA-FORMAT OPTIONAL,
        referenceTemplate    BioAPI-INPUT-BIR OPTIONAL,
        payload              BioAPI-DATA OPTIONAL,
        timeout              SignedInt,
        no-auditData         BOOLEAN,
        no-templateUuid      BOOLEAN
    }
```

и

```
Enroll-ResponseParams ::= SEQUENCE {
        newTemplate          BioAPI-BIR-HANDLE,
        auditData            BioAPI-BIR-HANDLE OPTIONAL,
        templateUuid         BioAPI-UUID OPTIONAL
    }
```

16.36.3 Когда структура получает вызов к функции **BioAPI_Enroll** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное

значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **enroll** согласно разделу 27, выполняя действия, указанные в 16.36.5 и 16.36.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.36.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **enroll** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Enroll** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **enroll** согласно разделу 28, выполняя действия, указанные в 16.36.5 и 16.36.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.36.5 Преобразование между параметрами функции Си **BioAPI_Enroll** и типом АСН.1 **Enroll-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 86.

Таблица 86 – Преобразование данных между параметрами функции **BioAPI_Enroll** и типом АСН.1 **Enroll-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
Purpose	purpose	15.14
Subtype	subtype	15.16
OutputFormat	outputFormat	Раздел 19 совместно с 15.46
ReferenceTemplate	referenceTemplate	Раздел 19 совместно с 15.46
NewTemplate	Отсутствует	Раздел 22
Payload	payload	Раздел 19 совместно с 15.22
Timeout	timeout	15.1.6
AuditData	no-auditData	Раздел 21
TemplateUuid	no-templateUuid	Раздел 21

16.36.6 Преобразование между параметрами функции Си **BioAPI_Enroll** и типом АСН.1 **Enroll-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 87.

Таблица 87 – Преобразование данных между параметрами функции **BioAPI_Enroll** и типом АСН.1 **Enroll-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
NewTemplate	newTemplate	Раздел 20 совместно с 15.12
AuditData	auditData	Раздел 20 совместно с 15.12
TemplateUuid	templateUuid	Раздел 20 совместно с 15.58

16.37 Функция BioAPI_Verify

16.37.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.37.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **Verify** и тип сообщения ответа **Verify**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
Verify-RequestParams ::= SEQUENCE {
                originalBSPHandle       BioAPI-HANDLE,
                maxFMRRequested         BioAPI-FMR,
                referenceTemplate       BioAPI-INPUT-BIR,
                subtype                 BioAPI-BIR-SUBTYPE,
                timeout                 SignedInt,
                no-adaptedBIR           BOOLEAN,
                no-fmrAchieved          BOOLEAN,
                no-payload              BOOLEAN,
                no-auditData            BOOLEAN
        }
```

и

```
Verify-ResponseParams ::= SEQUENCE {
                adaptedBIR              BioAPI-BIR-HANDLE OPTIONAL,
                result                  BOOLEAN,
                fmrAchieved             BioAPI-FMR OPTIONAL,
                payload                 BioAPI-DATA OPTIONAL,
                auditData               BioAPI-BIR-HANDLE OPTIONAL
        }
```

16.37.3 Когда структура получает вызов к функции **BioAPI_Verify** от локального приложения, она должна сначала определить главную конечную

точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **Verify** согласно разделу 27, выполняя действия, указанные в 16.37.5 и 16.37.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.37.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **Verify** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Verify** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **Verify** согласно разделу 28, выполняя действия, указанные в 16.37.5 и 16.37.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.37.5 Преобразование между параметрами функции Си **BioAPI_Verify** и типом АСН.1 **Verify-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 88.

Таблица 88 – Преобразование данных между параметрами функции **BioAPI_Verify** и типом ACH.1 **Verify-RequestParams**

Параметр функции	Компонент типа ACH.1	Ссылки
BSPHandle	originalBSPHandle	Раздел 26
MaxFMRRequested	maxFMRRequested	15.32
ReferenceTemplate	referenceTemplate	Раздел 19 совместно с 15.46
Subtype	subtype	15.16
AdaptedBIR	no-adaptedBIR	Раздел 21
Result	Отсутствует	Раздел 22
FMRAchieved	no-fmrAchieved	Раздел 21
Payload	no-payload	Раздел 21
Timeout	timeout	15.1.6
AuditData	no-auditData	Раздел 21

16.37.6 Преобразование между параметрами функции Си **BioAPI_Verify** и типом ACH.1 **Verify-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 89.

Таблица 89 – Преобразование данных между параметрами функции **BioAPI_Verify** и типом ACH.1 **Verify-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
AdaptedBIR	adaptedBIR	Раздел 20 совместно с 15.12
Result	result	Раздел 20 совместно с 15.18
FMRAchieved	fmrAchieved	Раздел 20 совместно с 15.32
Payload	payload	Раздел 20 совместно с 15.22
AuditData	auditData	Раздел 20 совместно с 15.12

16.38 Функция BioAPI_Identify

16.38.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.38.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **identify** и тип сообщения ответа **identify**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
Identify-RequestParams ::= SEQUENCE {
                originalBSPHandle           BioAPI-HANDLE,
                maxFMRRequested             BioAPI-FMR,
                subtype                     BioAPI-BIR-SUBTYPE,
                population                  BioAPI-IDENTIFY-POPULATION,
                totalNumberOfTemplates      UnsignedInt,
                binning                     BOOLEAN,
                maxNumberOfResults          UnsignedInt,
                timeout                     SignedInt,
                no-auditData                BOOLEAN
        }
```

и

```
Identify-ResponseParams ::= SEQUENCE {
                candidates                  SEQUENCE    (SIZE(0..max-unsigned-int)) OF
                                                    candidate BioAPI-CANDIDATE,
                auditData                   BioAPI-BIR-HANDLE OPTIONAL
        }
```

16.38.3 Когда структура получает вызов к функции **BioAPI_Identify** от локального приложения, она должна сначала определить главную конечную

точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **identify** согласно разделу 27, выполняя действия, указанные в 16.38.5 и 16.38.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.38.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **identify** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Identify** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **identify** согласно разделу 28, выполняя действия, указанные в 16.38.5 и 16.38.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.38.5 Преобразование между параметрами функции Си **BioAPI_Identify** и типом АСН.1 **Identify-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 90.

Таблица 90 – Преобразование данных между параметрами функции **BioAPI_Identify** и типом ACH.1 **Identify-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
MaxFMRRequested	maxFMRRequested	15.32
Subtype	subtype	15.16
Population	population	Раздел 19 совместно с 15.43
TotalNumberOfTemplates	totalNumberOfTemplates	15.1.5
Binning	binning	15.18
MaxNumberOfResults	maxNumberOfResults	15.1.5
NumberOfResults	Отсутствует	Раздел 22
Candidates	Отсутствует	Раздел 22
Timeout	timeout	15.1.6
AuditData	no-auditData	Раздел 21

16.38.6 Преобразование между параметрами функции Си **BioAPI_Identify** и типом ACH.1 **Identify-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 91.

Таблица 91 – Преобразование данных между параметрами функции **BioAPI_Identify** и типом ACH.1 **Identify-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
NumberOfResults, Candidates	candidates	Раздел 20 совместно с 16.38.7 и 16.38.8
AuditData	auditData	Раздел 20 совместно с 15.12

16.38.7 Преобразование двух переменных Си, выделенных параметрами **Candidates/NumberOfResults**, в компонент АСН.1 **candidates** выполняют следующим образом: принимают *N* равным значению переменной Си, выделенной параметром **NumberOfResults**; в этом случае первые элементы *N* типа **BioAPI_CANDIDATE** (см. 15.20) в массиве, выделенном переменной Си, которая выделена параметром **Candidates**, должны быть преобразованы по порядку в элемент компонента **candidates** согласно 15.20. Компонент **candidates** должен иметь точное число *N* элементов.

16.38.8 Преобразование компонента АСН.1 **candidates** в два переменных Си, выделенных параметрами **Candidates/NumberOfResults**, выполняют следующим образом: принимают *N* равным числу элементов компонента **candidates**, в этом случае новый массив *N* элементов типа **BioAPI_CANDIDATE** (см. 15.20) должен быть заполнен путем преобразования каждого элемента компонента **candidates** по порядку в элемент массива согласно 15.20. Переменная Си, выделенная параметром **Candidates** , должна быть установлена в адрес массива, а переменная Си, выделенная параметром **NumberOfResults**, должна быть установлена в *N*.

16.39 Функция BioAPI_Import

16.39.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.39.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **import** и тип сообщения ответа **import**,

которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
Import-RequestParams ::= SEQUENCE {
        originalBSPHandle BioAPI-HANDLE,
        inputData BioAPI-DATA,
        inputFormat BioAPI-BIR-BIOMETRIC-DATA-FORMAT,
        outputFormat BioAPI-BIR-BIOMETRIC-DATA-FORMAT OPTIONAL,
        purpose BioAPI-BIR-PURPOSE
    }
```

и

```
Import-ResponseParams ::= SEQUENCE {
        constructedBIR BioAPI-BIR-HANDLE
    }
```

16.39.3 Когда структура получает вызов к функции **BioAPI_Import** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **import** согласно разделу 27, выполняя действия, указанные в 16.39.5 и 16.39.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.39.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **import** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Import** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **import** согласно разделу 28, выполняя действия, указанные в 16.39.5 и 16.39.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.39.5 Преобразование между параметрами функции Си **BioAPI_Import** и типом АСН.1 **Import-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 92.

Таблица 92 – Преобразование данных между параметрами функции **BioAPI_Import** и типом АСН.1 **Import-RequestParams**

Параметр функции	Компонент типа АСН.1	Ссылки
BSPHandle	originalBSPHandle	Раздел 26
InputData	inputData	Раздел 19 совместно с 15.22
InputFormat	inputFormat	Раздел 19 совместно с 15.8
OutputFormat	outputFormat	Раздел 19 совместно с 15.8
Purpose	purpose	15.14
ConstructedBIR	Отсутствует	Раздел 22

16.39.6 Преобразование между параметрами функции Си **BioAPI_Import** и типом АСН.1 **Import-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 93.

Таблица 93 – Преобразование данных между параметрами функции **BioAPI_Import** и типом ACH.1 **Import-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
ConstructedBIR	constructedBIR	Раздел 20 совместно с 15.12

16.40 Функция BioAPI_PresetIdentifyPopulation

16.40.1 Данная функция определена в БиоАПИ следующим образом:

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

16.40.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **presetIdentifyPopulation** и тип сообщения ответа **presetIdentifyPopulation**, которые переносят значение следующего параметра типов ACH.1 сообщений ПМО БиоАПИ соответственно:

```
PresetIdentifyPopulation-RequestParams ::= SEQUENCE {
                originalBSPHandle           BioAPI-HANDLE,
                population                  BioAPI-IDENTIFY-POPULATION
        }
```

и

```
PresetIdentifyPopulation-ResponseParams ::= NULL
```

16.40.3 Когда структура получает вызов к функции **BioAPI_PresetIdentifyPopulation** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить

локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **presetIdentifyPopulation** согласно разделу 27, выполняя действия, указанные в 16.40.5 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.40.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **presetIdentifyPopulation** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_PresetIdentifyPopulation** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **presetIdentifyPopulation** согласно разделу 28, выполняя действия, указанные в 16.40.5 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.40.5 Преобразование между параметрами функции Си **BioAPI_PresetIdentifyPopulation** и типом АСН.1 **PresetIdentifyPopulation-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 94.

Таблица 94 – Преобразование данных между параметрами функции **BioAPI_PresetIdentifyPopulation** и типом АСН.1 **PresetIdentifyPopulation-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
Population	population	Раздел 19 совместно с 15.43

16.41 Функция BioAPI_Transform

16.41.1 Данная функция определена в БиоАПИ следующим образом:

```
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)
```

16.41.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **transform** и тип сообщения ответа **transform**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
Transform-RequestParams ::= SEQUENCE {
                bspHandle           BioAPI-HANDLE,
                operationUuid       BioAPI-UUID,
                inputBIRs           SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                        BioAPI-INPUT-BIR
        }
```

и

```
Transform-ResponseParams ::= SEQUENCE {
                outputBIRs          SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                        BioAPI-BIR-HANDLE
        }
```

16.41.3 Когда структура получает вызов к функции **BioAPI_Transform** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (originalBSPHandle) из параметра **BSPHandle** согласно разделу 26. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является

второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа трансформации ПМО БиоАПИ согласно разделу 27, выполняя действия, указанные в 16.41.5 и 16.41.8 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.41.4 Когда структура получает сообщение запроса трансформации ПМО БиоАПИ (см. 13.9) от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Transform** для создания и отправления соответствующего сообщения ответа трансформации ПМО БиоАПИ согласно разделу 28, выполняя действия, указанные в 16.41.5 и 16.41.8 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.41.5 Преобразование между параметрами функции Си **BioAPI_Transform** и типом АСН.1 **Transform-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 95.

Таблица 95 – Преобразование данных между параметрами функции **BioAPI_Transform** и типом АСН.1 **Transform-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
OperationUuid	operationUuid	15.58
InputBIRs, NumberOfInputBIRs	inputBIRs	16.41.6 и 16.41.7
OutputBIRs	Отсутствует	Раздел 22
NumberOfOutputBIRs	Отсутствует	Раздел 22

16.41.6 Преобразование двух параметров **InputBIRs/NumberOfInputBIRs** в компонент ACH.1 **inputBIRs** выполняют следующим образом: принимают *N* равным значению параметра **NumberOfInputBIRs**, каждый первый элемент *N* типа **BioAPI_INPUT_BIR** (см. 15.46) в массиве, выделенном параметром **InputBIRs**, должен быть преобразован по порядку в элемент компонента **inputBIRs** согласно 15.46.3. Компонент **inputBIRs** должен иметь точно число *N* элементов.

16.41.7 Преобразование компонента ACH.1 **inputBIRs** в два параметров **InputBIRs/NumberOfInputBIRs**, выполняют следующим образом: принимают *N* равным числу элементов компонента **guiBitmaps**, в этом случае новый массив *N* элементов типа **BioAPI_INPUT_BIR** (см. 15.46) должен быть заполнен путем преобразования каждого элемента компонента **inputBIRs** по порядку в элемент массива согласно 15.46.4. Параметр **InputBIRs** должен быть установлен в адрес массива, а параметр **NumberOfInputBIRs** должен быть установлен в *N*.

16.41.8 Преобразование между параметрами функции Си **BioAPI_Transform** и типом ACH.1 **Transform-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 96.

Таблица 96 – Преобразование данных между параметрами функции **BioAPI_Transform** и типом ACH.1 **Transform-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
OutputBIRs, NumberOfOuputBIRs	ouputBIRs	Раздел 29 совместно с 16.41.9 и 16.41.10

16.41.9 Преобразование двух переменных СИ, выделенных параметрами **OutputBIRs/NumberOfOutputBIRs**, в компонент ACH.1 **outputBIRs** выполняют следующим образом: принимают *N* равным значению

переменной Си, выделенной параметром **NumberOfOutputBIRs**; в этом случае первые элементы *N* типа **BioAPI_BIR_HANDLE** (см. 15.12) в массиве, выделенном переменной Си, которая выделена параметром **OutputBIRs**, должны быть преобразованы по порядку в элемент компонента **outputBIRs** согласно 15.46.3. Компонент **outputBIRs** должен иметь точное число *N* элементов.

16.41.10 Преобразование компонента ACH.1 **outputBIRs** в две переменные Си, выделенных параметрами **OutputBIRs/NumberOfOutputBIRs**, выполняют следующим образом: принимают *N* равным числу элементов компонента **outputBIRs**, в этом случае новый массив *N* элементов типа **BioAPI_BIR_HANDLE** (см. 15.12) должен быть заполнен путем преобразования каждого элемента компонента **outputBIRs** по порядку в элемент массива согласно 15.46.4. Переменная Си, выделенная параметром **OutputBIRs** , должна быть установлена в адрес массива, а переменная Си, выделенная параметром **NumberOfOutputBIRs**, должна быть установлена в *N*.

16.42 Функция BioAPI_DbOpen

16.42.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.42.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbOpen** и тип сообщения ответа **dbOpen**, которые переносят значение следующего параметра типов ACH.1 сообщений ПМО БиоАПИ соответственно:

```
DbOpen-RequestParams ::= SEQUENCE {
                originalBSPHandle BioAPI-HANDLE,
```

```
            dbUuid BioAPI-UUID,
            accessRequest BioAPI-DB-ACCESS-TYPE
        }
```

и

```
    DbOpen-ResponseParams ::= SEQUENCE {
            dbHandle BioAPI-DB-HANDLE,
            markerHandle BioAPI-DB-MARKER-HANDLE
        }
```

16.42.3 Когда структура получает вызов к функции **BioAPI_DbOpen** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbOpen** согласно разделу 27, выполняя действия, указанные в 16.42.5 и 16.42.6 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.42.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbOpen** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbOpen** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbOpen** согласно разделу 28, выполняя действия, указанные в 16.42.5 и 16.42.6, для

преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.42.5 Преобразование между параметрами функции Си **BioAPI_DbOpen** и типом АСН.1 **DbOpen-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 97.

Таблица 97 – Преобразование данных между параметрами функции **BioAPI_DbOpen** и типом АСН.1 **DbOpen-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
DbUuid	dbUuid	Раздел 19 совместно с 15.58
AccessRequest	accessRequest	15.24
DbHandle	Отсутствует	Раздел 22
MarkerHandle	Отсутствует	Раздел 22

16.42.6 Преобразование между параметрами функции Си **BioAPI_DbOpen** и типом АСН.1 **DbOpen-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 98.

Таблица 98 – Преобразования данных между параметрами функции **BioAPI_DbOpen** и типом АСН.1 **DbOpen-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
DbHandle	dbHandle	Раздел 20 совместно с 15.26
MarkerHandle	marker	Раздел 20 совместно с 15.25

16.43 Функция BioAPI_DbClose

16.43.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_DbClose
        (BioAPI_HANDLE BSPHandle,
        BioAPI_DB_HANDLE DbHandle);
```

16.43.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbClose** и тип сообщения ответа **dbClose**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ (соответственно):

```
DbClose-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                dbHandle             BioAPI-DB-HANDLE
        }
```

и

```
DbClose-ResponseParams ::= NULL
```

16.43.3 Когда структура получает вызов к функции **BioAPI_DbClose** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbClose** согласно разделу 27, выполняя действия, указанные в 16.43.5 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.43.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbClose** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbClose** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbClose** согласно разделу 28, выполняя действия, указанные в 16.43.5 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.43.5 Преобразование между параметрами функции Си **BioAPI_DbClose** и типом АСН.1 **DbClose- RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 99.

Таблица 99 – Преобразование данных между параметрами функции **BioAPI_DbClose** и типом АСН.1 **DbClose-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздел настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
DbHandle	dbHandle	15.26

16.44 Функция BioAPI_DbCreate

16.44.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_DbCreate
        (BioAPI_HANDLE BSPHandle,
        const BioAPI_UUID *DbUuid,
        uint32_t NumberOfRecords,
        BioAPI_DB_ACCESS_TYPE AccessRequest,
        BioAPI_DB_HANDLE *DbHandle);
```

16.44.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbCreate** и тип сообщения ответа

dbCreate, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
DbCreate-RequestParams ::= SEQUENCE {
        originalBSPHandle    BioAPI-HANDLE,
        dbUuid               BioAPI-UUID,
        numberOfRecords      UnsignedInt,
        accessRequest        BioAPI-DB-ACCESS-TYPE
    }
```

и

```
DbCreate-ResponseParams ::= SEQUENCE {
        dbHandle             BioAPI-DB-HANDLE
    }
```

16.44.3 Когда структура получает вызов к функции **BioAPI_DbCreate** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbCreate** согласно разделу 27, выполняя действия, указанные в 16.44.5 и 16.44.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.44.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbCreate** от главной конечной точки, она должна обработать запрос

путем внутреннего вызова функции БиоАПИ к **BioAPI_DbCreate** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbCreate** согласно разделу 28, выполняя действия, указанные в 16.44.5 и 16.44.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.44.5 Преобразование между параметрами функции Си **BioAPI_DbCreate** и типом АСН.1 **DbCreate-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 100.

Таблица 100 – Преобразование данных между параметрами функции **BioAPI_DbCreate** и типом АСН.1 **DbCreate-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего станадарта
BSPHandle	originalBSPHandle	Раздел 26
DbUuid	dbUuid	Раздел 19 совместно с 15.58
NumberOfRecords	numberOfRecords	15.1.5
AccessRequest	accessRequest	15.24
DbHandle	Отсутствует	Раздел 22

16.44.6 Преобразование между параметрами функции Си **BioAPI_DbCreate** и типом АСН.1 **DbCreate- ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 101.

Таблица 101 – Преобразования данных между параметрами функции **BioAPI_DbCreate** и типом АСН.1 **DbCreate-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздел настоящего стандарта
DbHandle	dbHandle	Раздел 20 совместно с 15.26

16.45 Функция BioAPI_DbDelete

16.45.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_DbDelete
        (BioAPI_HANDLE BSPHandle,
        const BioAPI_UUID *DbUuid);
```

16.45.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbDelete** и тип сообщения ответа **dbDelete**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
DbDelete-RequestParams ::= SEQUENCE {
                originalBSPHandle     BioAPI-HANDLE,
                dbUuid                BioAPI-UUID
        }
```

и

```
DbDelete-ResponseParams ::= NULL
```

16.45.3 Когда структура получает вызов к функции **BioAPI_DbDelete** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 13.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является

второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbDelete** согласно разделу 27, выполняя действия, указанные в 16.45.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.45.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbDelete** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbDelete** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbDelete** согласно разделу 28, выполняя действия, указанные в 16.45.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.45.5 Преобразование между параметрами функции Си **BioAPI_DbDelete** и типом АСН.1 **DbDelete-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 102.

Таблица 102 – Преобразование данных между параметрами функции **BioAPI_DbDelete** и типом АСН.1 **DbDelete-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
DbUuid	dbUuid	Раздел 19 совместно с 15.58

16.46 Функция BioAPI_DbSetMarker

16.46.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_DbSetMarker
        (BioAPI_HANDLE BSPHandle,
```

```
BioAPI_DB_HANDLE DbHandle,
const BioAPI_UUID *KeyValue,
BioAPI_DB_MARKER_HANDLE MarkerHandle);
```

16.46.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbSetMarker** и тип сообщения ответа **dbSetMarker**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
DbSetMarker-RequestParams ::= SEQUENCE {
        originalBSPHandle   BioAPI-HANDLE,
        dbHandle            BioAPI-DB-HANDLE,
        keyValue            BioAPI-UUID,
        markerHandle        BioAPI-DB-MARKER-HANDLE
    }
```

и

```
DbSetMarker-ResponseParams ::= NULL
```

16.46.3 Когда структура получает вызов к функции **BioAPI_DbSetMarker** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbSetMarker** согласно разделу 27, выполняя действия, указанные в 16.46.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена,

структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.46.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbSetMarker** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbSetMarker** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbSetMarker** согласно разделу 28, выполняя действия, указанные в 16.46.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.46.5 Преобразование между параметрами функции Си **BioAPI_DbSetMarker** и типом АСН.1 **DbSetMarker-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 103.

Таблица 103 – Преобразование данных между параметрами функции **BioAPI_DbSetMarker** и типом АСН.1 **DbSetMarker-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
DbHandle	dbHandle	15.26
KeyValue	keyValue	Раздел 19 совместно с 15.58
MarkerHandle	markerHandle	15.25

16.47 Функция BioAPI_DbFreeMarker

16.47.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_DbFreeMarker
    (BioAPI_HANDLE BSPHandle,
    BioAPI_DB_MARKER_HANDLE MarkerHandle);
```

16.47.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbFreeMarker** и тип сообщения ответа **dbFreeMarker**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
DbFreeMarker-RequestParams ::= SEQUENCE {
        originalBSPHandle    BioAPI-HANDLE,
        markerHandle         BioAPI-DB-MARKER-HANDLE
    }
```

и

```
DbFreeMarker-ResponseParams ::= NULL
```

16.47.3 Когда структура получает вызов к функции **BioAPI_DbFreeMarker** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbFreeMarker** согласно разделу 27, выполняя действия, указанные в 16.47.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.47.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbFreeMarker** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbFreeMarker** для

создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbFreeMarker** согласно разделу 28, выполняя действия, указанные в 16.47.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.47.5 Преобразование между параметрами функции Си **BioAPI_DbFreeMarker** и типом АСН.1 **DbFreeMarker-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 104.

Таблица 104 – Преобразование данных между параметрами функции **BioAPI_DbFreeMarker** и типом АСН.1 **DbFreeMarker-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
MarkerHandle	markerHandle	15.25

16.48 Функция BioAPI_DbStoreBIR

16.48.1Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_DbStoreBIR
        (BioAPI_HANDLE BSPHandle,
        const BioAPI_INPUT_BIR *BIRToStore,
        BioAPI_DB_HANDLE DbHandle,
        BioAPI_UUID *BirUuid);
```

16.48.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbStoreBIR** и тип сообщения ответа **dbStoreBIR**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
DbStoreBIR-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                birToStore           BioAPI-INPUT-BIR,
                dbHandle             BioAPI-DB-HANDLE
        }
```

и

```
DbStoreBIR-ResponseParams ::= SEQUENCE {
        birUuid                 BioAPI-UUID
}
```

16.48.3 Когда структура получает вызов к функции **BioAPI_DbStoreBIR** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbStoreBIR** согласно разделу 27, выполняя действия, указанные в 16.48.5 и 16.48.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.48.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbStoreBIR** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbStoreBIR** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbStoreBIR** согласно разделу 28, выполняя действия, указанные в 16.48.5 и 16.48.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.48.5 Преобразование между параметрами функции Си **BioAPI_DbStoreBIR** и типом АСН.1 **DbStoreBIR-RequestParams**

выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 105.

Таблица 105 – Преобразование данных между параметрами функции **BioAPI_DbStoreBIR** и типом АСН.1 **DbStoreBIR-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
BIRToStore	birToStore	Раздел 19 совместно с 15.46
DbHandle	dbHandle	15.26
BirUuid	Отсутствует	Раздел 22

16.48.6 Преобразование между параметрами функции Си **BioAPI_DbStoreBIR** и типом АСН.1 **DbStoreBIR-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 106.

Таблица 106 – Преобразование данных между параметрами функции **BioAPI_DbStoreBIR** и типом АСН.1 **DbStoreBIR-ResponseParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BirUuid	birUuid	раздел 20 совместно с 15.58

16.49 Функция BioAPI_DbGetBIR

16.49.1 Данная функция определена в БиоАПИ следующим образом:

```
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);
```

16.49.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbGetBIR** и тип сообщения ответа **dbGetBIR**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
DbGetBIR-RequestParams ::= SEQUENCE {
        originalBSPHandle    BioAPI-HANDLE,
        dbHandle             BioAPI-DB-HANDLE,
        keyValue             BioAPI-UUID
    }
```

и

```
DbGetBIR-ResponseParams ::= SEQUENCE {
        retrievedBIR         BioAPI-BIR-HANDLE,
        markerHandle         BioAPI-DB-MARKER-HANDLE
    }
```

16.49.3 Когда структура получает вызов к функции **BioAPI_DbGetBIR** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точкая, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbGetBIR** согласно разделу 27, выполняя действия, указанные в 16.49.5 и 16.49.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.49.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbGetBIR** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbGetBIR** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbGetBIR** согласно разделу 28, выполняя действия, указанные в 16.49.5 и 16.49.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в данном разделе.

16.49.5 Преобразование между параметрами функции Си **BioAPI_DbGetBIR** и типом АСН.1 **DbGetBIR-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 107.

Таблица 107 – Преобразование данных между параметрами функции **BioAPI_DbGetBIR** и типом АСН.1 **DbGetBIR-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
DbHandle	dbHandle	15.26
KeyValue	keyValue	Раздел 19 совместно с 15.58
RetrievedBIR	Отсутствует	Раздел 22
MarkerHandle	Отсутствует	Раздел 22

16.49.6 Преобразование между параметрами функции Си **BioAPI_DbGetBIR** и типом АСН.1 **DbGetBIR-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 108.

Таблица 108 – Преобразование данных между параметрами функции **BioAPI_DbGetBIR** и типом ACH.1 **DbGetBIR-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
RetrievedBIR	retrievedBIR	Раздел 20 совместно с 15.12
MarkerHandle	markerHandle	Раздел 20 совместно с 15.25

16.50 Функция BioAPI_DbGetNextBIR

16.50.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_DbGetNextBIR
        (BioAPI_HANDLE BSPHandle,
        BioAPI_DB_HANDLE DbHandle,
        BioAPI_DB_MARKER_HANDLE MarkerHandle,
        BioAPI_BIR_HANDLE *RetrievedBIR,
        BioAPI_UUID *BirUuid);
```

16.50.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbGetNextBIR** и тип сообщения ответа **dbGetNextBIR**, которые переносят значение следующего параметра типов ACH.1 сообщений ПМО БиоАПИ соответственно:

```
DbGetNextBIR-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                dbHandle             BioAPI-DB-HANDLE,
                markerHandle         BioAPI-DB-MARKER-HANDLE
        }
```

и

```
DbGetNextBIR-ResponseParams ::= SEQUENCE {
                retrievedBIR         BioAPI-BIR-HANDLE,
                birUuid                      BioAPI-UUID
        }
```

16.50.3 Когда структура получает вызов к функции **BioAPI_DbGetNextBIR** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ

(*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbGetNextBIR** согласно разделу 27, выполняя действия, указанные в 16.50.5 и 16.50.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.50.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbGetNextBIR** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbGetNextBIR** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **dbGetNextBIR** согласно разделу 28, выполняя действия, указанные в 16.50.5 и 16.50.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.50.5 Преобразование между параметрами функции Си **BioAPI_DbGetNextBIR** и типом АСН.1 **DbGetNextBIR-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 109.

Таблица 109 – Преобразование данных между параметрами функции **BioAPI_DbGetNextBIR** и типом ACH.1 **DbGetNextBIR-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
DbHandle	dbHandle	15.26
MarkerHandle	markerHandle	15.25
RetrievedBIR	Отсутствует	Раздел 22
BirUuid	Отсутствует	Раздел 22

16.50.6 Преобразование между параметрами функции Си **BioAPI_DbGetNextBIR** и типом ACH.1 **DbGetNextBIR-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 110.

Таблица 110 – Преобразование данных между параметрами функции **BioAPI_DbGetNextBIR** и типом ACH.1 **DbGetNextBIR-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
RetrievedBIR	retrievedBIR	Раздел 20 совместно с 15.12
BirUuid	birUuid	Раздел 20 совместно с 15.58

16.51 Функция BioAPI_DbDeleteBIR

16.51.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_DbDeleteBIR
        (BioAPI_HANDLE BSPHandle,
        BioAPI_DB_HANDLE DbHandle,
        const BioAPI_UUID *KeyValue);
```

16.51.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **dbDeleteBIR** и тип сообщения ответа

dbDeleteBIR, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
DbDeleteBIR-RequestParams ::= SEQUENCE {
        originalBSPHandle    BioAPI-HANDLE,
        dbHandle             BioAPI-DB-HANDLE,
        keyValue             BioAPI-UUID
    }
```

и

```
DbDeleteBIR-ResponseParams ::= NULL
```

16.51.3 Когда структура получает вызов к функции **BioAPI_DbDeleteBIR** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **dbDeleteBIR** согласно разделу 27, выполняя действия, указанные в 16.51.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.51.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **dbDeleteBIR** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_DbDeleteBIR** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ

dbDeleteBIR согласно разделу 28, выполняя действия, указанные в 16.51.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.51.5 Преобразование между параметрами функции Си **BioAPI_DbDeleteBIR** и типом АСН.1 **DbDeleteBIR-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 111.

Таблица 111 – Преобразование данных между параметрами функции **BioAPI_DbDeleteBIR** и типом АСН.1 **DbDeleteBIR-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
DbHandle	dbHandle	15.26
KeyValue	keyValue	Раздел 19 совместно с 15.58

16.52 Функция BioAPI_CalibrateSensor

16.52.1Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_CalibrateSensor
        (BioAPI_HANDLE BSPHandle,
        int32_t Timeout);
```

16.52.2С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **calibrateSensor** и тип сообщения ответа **calibrateSensor**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
CalibrateSensor-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                timeout              SignedInt
        }
```

и

```
CalibrateSensor-ResponseParams ::= NULL
```

16.52.3 Когда структура получает вызов к функции **BioAPI_CalibrateSensor** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **calibrateSensor** согласно разделу 27, выполняя действия, указанные в 16.52.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.52.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **calibrateSensor** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_CalibrateSensor** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **calibrateSensor** согласно разделу 28, выполняя действия, указанные в 16.52.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.52.5 Преобразование между параметрами функции Си **BioAPI_CalibrateSensor** и типом АСН.1 **CalibrateSensor-RequestParams**

выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 112.

Таблица 112 – Преобразование данных между параметрами функции **BioAPI_CalibrateSensor** и типом АСН.1 **CalibrateSensor-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, подраздел настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
Timeout	timeout	15.1.6

16.53 Функция BioAPI_SetPowerMode

16.53.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_SetPowerMode
        (BioAPI_HANDLE BSPHandle,
        BioAPI_UNIT_ID UnitID,
        BioAPI_POWER_MODE PowerMode);
```

16.53.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **setPowerMode** и тип сообщения ответа **setPowerMode**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
SetPowerMode-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                unitID               BioAPI-UNIT-ID,
                powerMode            BioAPI-POWER-MODE
        }
```

и

```
SetPowerMode-ResponseParams ::= NULL
```

16.53.3 Когда структура получает вызов к функции **BioAPI_SetPowerMode** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. В случае, если главной конечной точкой является локальная конечная точка, структура

должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **setPowerMode** согласно разделу 27, выполняя действия, указанные в 16.53.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.53.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **setPowerMode** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_SetPowerMode** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **setPowerMode** согласно разделу 28, выполняя действия, указанные в 16.53.5 для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.53.5 Преобразование между параметрами функции Си **BioAPI_SetPowerMode** и типом АСН.1 **SetPowerMode-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 113.

Таблица 113 – Преобразование данных между параметрами функции **BioAPI_SetPowerMode** и типом АСН.1 **SetPowerMode-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
UnitID	unitID	15.55
PowerMode	powerMode	15.50

16.54 Функция BioAPI_SetIndicatorStatus

16.54.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_SetIndicatorStatus
        (BioAPI_HANDLE BSPHandle,
        BioAPI_UNIT_ID UnitID,
        BioAPI_INDICATOR_STATUS IndicatorStatus);
```

16.54.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **setIndicatorStatus** и тип сообщения ответа **setIndicatorStatus**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
SetIndicatorStatus-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                unitID               BioAPI-UNIT-ID,
                indicatorStatus      BioAPI-INDICATOR-STATUS
        }
```

и

```
SetIndicatorStatus-ResponseParams ::= NULL
```

16.54.3 Когда структура получает вызов к функции **BioAPI_SetIndicatorStatus** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением

параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **setIndicatorStatus** согласно разделу 27, выполняя действия, указанные в 16.54.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.54.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **setIndicatorStatus** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_SetIndicatorStatus** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **setIndicatorStatus** согласно разделу 28, выполняя действия, указанные в 16.54.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.54.5 Преобразование между параметрами функции Си **BioAPI_SetIndicatorStatus** и типом АСН.1 **SetIndicatorStatus-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 114.

Таблица 114 – Преобразование данных между параметрами функции **BioAPI_SetIndicatorStatus** и типом АСН.1 **SetIndicatorStatus-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
UnitID	unitID	15.55
IndicatorStatus	indicatorStatus	15.45

16.55 Функция BioAPI_GetIndicatorStatus

16.55.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_GetIndicatorStatus
        (BioAPI_HANDLE BSPHandle,
        BioAPI_UNIT_ID UnitID,
        BioAPI_INDICATOR_STATUS *IndicatorStatus);
```

16.55.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **getIndicatorStatus** и тип сообщения ответа **getIndicatorStatus**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
GetIndicatorStatus-RequestParams ::= SEQUENCE {
                originalBSPHandle    BioAPI-HANDLE,
                unitID               BioAPI-UNIT-ID
        }
```

и

```
GetIndicatorStatus-ResponseParams ::= SEQUENCE {
                indicatorStatus              BioAPI-INDICATOR-STATUS
        }
```

16.55.3 Когда структура получает вызов к функции **BioAPI_GetIndicatorStatus** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна

выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**, который должен быть установлен путем преобразования *originalBSPHandle* согласно 14.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главная конечная точка является второстепенной конечной точкой структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **getIndicatorStatus** согласно разделу 27, выполняя действия, указанные в 16.55.5 и 16.55.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.55.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **getIndicatorStatus** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_GetIndicatorStatus** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **getIndicatorStatus** согласно разделу 28, выполняя действия, указанные в 16.55.5 и 16.55.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.55.5 Преобразование между параметрами функции Си **BioAPI_GetIndicatorStatus** и типом АСН.1 **GetIndicatorStatus-RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 115.

Таблица 115 – Преобразование данных между параметрами функции **BioAPI_GetIndicatorStatus** и типом ACH.1 **GetIndicatorStatus-RequestParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26
UnitID	unitID	15.55
IndicatorStatus	Отсутствует	Раздел 22

16.55.6 Преобразование между параметрами функции Си **BioAPI_GetIndicatorStatus** и типом ACH.1 **GetIndicatorStatus-ResponseParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 116.

Таблица 116 – Преобразование данных между параметрами функции **BioAPI_GetIndicatorStatus** и типом ACH.1 **GetIndicatorStatus-ResponseParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
IndicatorStatus	indicatorStatus	Раздел 20 совместно с 15.45

16.56 Функция BioAPI_GetLastErrorInfo

16.56.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_GetLastErrorInfo
    (BioAPI_ERROR_INFO *ErrorInfo);
```

16.56.2 Связанные типы сообщений ПМО БиоАПИ отсутствуют.

16.56.3 Когда структура получает вызов к функции **BioAPI_GetLastErrorInfo** от локального приложения, она должна определить члены исходящего параметра **ErrorInfo** (см. 15.29) с информацией о самом

последнем состоянии ошибки, которое привело к возвращению кода ошибки функцией БиоАПИ. Если локальное приложение выполняет вызов функций БиоАПИ по нескольким потокам, структура возвращает информацию о последней произошедшей ошибке в функции БиоАПИ, вызываемой по тому же потоку от приложения, по которому она была вызвана.

16.56.4 Данная функция главным образом предназначена для предоставления диагностической информации локальному приложению. Так как возможные значения членов **ErrorInfo** не стандартизированы и могут различаться на различных платформах и реализациях, обычные приложения БиоАПИ не должны полагаться на такие значения в любых случаях принятия важных решений обработки.

16.57 Функция BioAPI_Cancel

16.57.1 Данная функция определена в БиоАПИ следующим образом:

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

16.57.2 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **cancel** и тип сообщения ответа **cancel**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
Cancel-RequestParams ::= SEQUENCE {
                originalBSPHandle     BioAPI-HANDLE
        }
```

и

```
Cancel-ResponseParams ::= NULL
```

16.57.3 Когда структура получает вызов к функции **BioAPI_Cancel** от локального приложения, она должна сначала определить главную конечную точку и исходный обработчик ПБУ (*originalBSPHandle*) из параметра **BSPHandle** согласно разделу 24. Если главной конечной точкой является локальная конечная точка, структура должна выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, за исключением параметра **BSPHandle**,

который должен быть установлен путем преобразования *originalBSPHandle* согласно 15.42, а также возвратить локальному приложению возвращенное значение внутреннего вызова. Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **cancel** согласно разделу 27, выполняя действия, указанные в 16.57.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе. Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.57.4 Когда структура получает сообщение запроса ПМО БиоАПИ (см. 13.9) **cancel** от главной конечной точки, она должна обработать запрос путем внутреннего вызова функции БиоАПИ к **BioAPI_Cancel** для создания и отправления соответствующего сообщения ответа ПМО БиоАПИ **cancel** как указано в разделе 28, выполняя действия, указанные в 16.57.5, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.57.5 Преобразование между параметрами функции Си **BioAPI_Cancel** и типом АСН.1 **Cancel- RequestParams** выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 117.

Таблица 117 – Преобразование данных между параметрами функции **BioAPI_Cancel** и типом АСН.1 **Cancel-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел настоящего стандарта
BSPHandle	originalBSPHandle	Раздел 26

16.58 Функция BioAPI_Free

16.58.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI BioAPI_Free
    (void *Ptr);
```

16.58.2 Связанные типы сообщений ПМО БиоАПИ отсутствуют.

16.58.3 Когда структура получает вызов к функции **BioAPI_Free** от локального приложения, она должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **ApplicationOwnedMemoryBlocks** (см. 18.13) на наличие поля, в котором компонент **address** имеет такое же значение, как и параметр **Ptr**;

b) в случае, если такое поле не обнаружено, выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как во входящем вызове, и возвратить локальному приложению возвращенное значение внутреннего вызова без выполнения следующих действий;

c) очистить соответствующий блок памяти;

d) удалить поле таблицы **ApplicationOwnedMemoryBlocks**;

e) возвратить значение 0 локальному приложению.

16.59 Функция BioAPI_RegisterBSP

16.59.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI_RegisterBSP
    (const uint8_t *HostingEndpointIRI,
    const BioAPI_BSP_SCHEMA *BSPSchema,
    BioAPI_BOOL Update);
```

16.59.2 С данной функцией связаны три типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **registerBSP**, тип сообщения ответа **registerBSP** и тип сообщения уведомления ПМО БиоАПИ **bspRegistrationEvent**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
RegisterBSP-RequestParams ::= SEQUENCE {
```

```
            bspSchema     BioAPI-BSP-SCHEMA,
            update BOOLEAN
        }

    RegisterBSP-ResponseParams ::= NULL
```

и

```
    BSPRegistrationEvent-NotificationParams ::= SEQUENCE {
            bspSchema     BioAPI-BSP-SCHEMA,
            update BOOLEAN
        }
```

16.59.3 Когда структура получает вызов к функции **BioAPI_RegisterBSP** от локального приложения, она сначала определяет ИИР главной конечной точки путем преобразования из параметра **HostingEndpointIRI** согласно 15.3, и только затем выполняют действия, указанные в одном из следующих подпунктов.

16.59.3.1 Если главной конечной точкой является локальная конечная точка, структура должна выполнить следующие действия в указанном порядке:

a) совершить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как и во входящем вызове;

b) в случае, если возвращенное значение внутреннего вызова не является 0, возвратить такое значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*incomingRequestParams*) типа **RegisterBSPRequestParams** (см. 16.59.2) путем преобразования из параметров вызова функции **BioAPI_RegisterBSP** согласно 16.59.6;

d) проверить таблицу **VisibleBSPRegistrations** (см. 18.3) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки и

2) компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** компонента **bspSchema** *incomingRequestParams*;

e) в случае, если такое поле обнаружено и компонент **update** *incomingRequestParams* имеет значение **FALSE**, возвратить значение **BioAPIERR_COMPONENT_ALREADY_REGISTERED** локальному приложению без выполнения следующих действий;

f) Если такое поле обнаружено, а компонент **update** *incomingRequestParams* имеет значение **TRUE**, удалить поле таблицы **VisibleBSPRegistrations** выполняя действия, указанные в 18.3.3;

g) добавить поле в таблицу **VisibleBSPRegistrations** (см. 18.3), в котором:

1) компонент **hostingEndpointIRI** установлен на ИИР локальной конечной точки;

2) компонент **bspAccessUuid** установлен на динамически созданный УУИД; и

3) оставшиеся компоненты установлены из компонентов компонента **bspSchema** *incomingRequestParams* с такими же именами;

h) создать временное абстрактное значение (*outgoingNotificationParams*) типа **BSPRegistrationEvent- NotificationParams** (см. 16.59.2), в котором все компоненты установлены из компонентов *incomingRequestParams* с такими же именами;

i) для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1) создать и отправить сообщение уведомления ПМО БиоАПИ **bspRegistrationEvent** (см. 13.4) с ИИР главной конечной точки, установленным из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра, установленным на *outgoingNotificationParams*;

j) возвратить значение 0 локальному приложению.

16.59.3.2 Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **registerBSP** согласно разделу 27, выполняя действия, указанные в 16.59.6, для

преобразования между параметрами функции и компонентами ACH.1, если это установлено в указанном разделе.

16.59.3.3 Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_BSP** локальному приложению.

16.59.4 Когда структура принимает сообщение запроса ПМО БиоАПИ (см. 13.9) **registerBSP** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **RegisterBSP-RequestParams** (см. 16.59.2) сообщения запроса ПМО БиоАПИ **registerBSP**;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_RegisterBSP**, в котором параметры вызова функции установлены путем преобразований из *incomingRequestParams* согласно 16.59.6;

c) Если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **registerBSP** (см. 13.3) с возвращаемым значением, установленным на это значение без выполнения следующих действий;

d) проверить таблицу **VisibleBSPRegistrations** (см. 18.3) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки и

2) компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** компонента **bspSchema** *incomingRequestParams*;

e) Если такое поле обнаружено и компонент **update** *incomingRequestParams* имеет значение **FALSE**, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **registerBSP** (см.

13.3) с возвращаемым значением, установленным на **BioAPIERR_COMPONENT_ALREADY_REGISTERED** без выполнения следующих действий;

f) Если такое поле обнаружено, а компонент **update** *incomingRequestParams* имеет значение **TRUE**, удалить поле таблицы **VisibleBSPRegistrations** выполняя действия, указанные в 18.3.3;

g) добавить поле в таблицу **VisibleBSPRegistrations** (см. 18.3), в котором:

1) компонент **hostingEndpointIRI** установлен на ИИР локальной конечной точки;

2) компонент **bspAccessUuid** установлен на динамически созданный УУИД; и

3) оставшиеся компоненты должны быть установлены из компонентов компонента **bspSchema** *incomingRequestParams* с такими же именами;

h) для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1) создать и отправить сообщение уведомления ПМО БиоАПИ **bspRegistrationEvent** (см. 13.4) с ИИР главной конечной точки, установленным из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра, установленным на *incomingRequestParams*;

i) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **registerBSP** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращаемым значением, установленным на 0.

16.59.5 Когда структура принимает сообщение уведомления ПМО БиоАПИ (см. 13.9) **bspRegistrationEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingNotificationParams* выступать в качестве значения параметра типа **BSPRegistrationEvent-NotificationParams** (см. 16.59.2) сообщения уведомления ПМО БиоАПИ **bspRegistrationEvent**;

b) проверить таблицу **VisibleBSPRegistrations** (см. 18.3) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР второстепенной конечной точки и

2) компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** компонента **bspSchema** *incomingNotificationParams*;

c) в случае, если такое поле обнаружено и компонент **update** *incomingNotificationParams* имеет значение **FALSE**, не выполнять следующие действия;

d) в случае, если такое поле обнаружено и компонент **update** *incomingNotificationParams* имеет значение **TRUE**, удалить поле таблицы **VisibleBSPRegistrations**, выполняя действия, указанные в 18.3.3;

e) добавить поле в таблицу **VisibleBSPRegistrations** (см. 18.3), в котором:

1) компонент **bspAccessUuid** установлен на динамически созданный УУИД; и

2) оставшиеся компоненты установлены из компонентов компонента **bspSchema** *incomingNotificationParams* с такими же именами;

f) заключить, что сообщение подтверждения ПМО БиоАПИ для отправления отсутствует.

16.59.6 Преобразование между параметрами функции Си **BioAPI_RegisterBSP** и типом АСН.1 **RegisterBSP-RequestParams** (см. 16.59.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 118.

Таблица 118 – Преобразование данных между параметрами функции **BioAPI_RegisterBSP** и типом АСН.1 **RegisterBSP-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
HostingEndpointIRI	Отсутствует	16.59.7
BSPSchema	bspSchema	Раздел 19 совместно с 15.19
Update	update	15.18

16.59.7 При преобразовании из параметров функции Си в тип АСН.1, параметр **HostingEndpointIRI** должен быть проигнорирован, так как он уже использовался для определения главной конечной точки. При преобразовании из типа АСН.1 в параметры функции Си, параметр **HostingEndpointIRI** должен был установлен на **NULL**, для определения локальной конечной точки.

16.60 Функция BioAPI_UnregisterBSP

16.60.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI_UnregisterBSP
        (const uint8_t *HostingEndpointIRI,
        const BioAPI_UUID *BSPProductUuid);
```

16.60.2 С данной функцией связаны три типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **unregisterBSP**, тип сообщения ответа **unregisterBSP** и тип сообщения уведомления ПМО БиоАПИ **bspUnregistrationEvent**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
UnregisterBSP-RequestParams ::= SEQUENCE {
                bspProductUuid      BioAPI-UUID
        }

UnregisterBSP-ResponseParams ::= NULL
```

и

```
BSPUnregistrationEvent-NotificationParams ::= SEQUENCE {
                bspProductUuid      BioAPI-UUID
        }
```

16.60.3 Когда структура получает вызов к функции **BioAPI_UnregisterBSP** от локального приложения, она сначала определяет ИИР главной конечной точки путем преобразования из параметра **HostingEndpointIRI** согласно 15.3, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.60.3.1 Если главной конечной точкой является локальная конечная точка, структура должна выполнить следующие действия в указанном порядке:

a) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как и во входящем вызове;

b) в случае, если возвращенное значение внутреннего вызова не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*incomingRequestParams*) типа **UnregisterBSPRequestParams** (см. 16.60.2) путем преобразования параметров вызова функции **BioAPI_UnregisterBSP** согласно 16.60.6;

d) проверить таблицу **VisibleBSPRegistrations** (см. 18.3) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки и

2) компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** *incomingRequestParams*;

e) в случае, если такое поле не обнаружено, возвратить значение **BioAPIERR_NO_SUCH_COMPONENT_FOUND** локальному приложению без выполнения следующих действий;

f) удалить поле таблицы **VisibleBSPRegistrations** (выполняя действия, указанные в 18.3.3);

g) создать временное абстрактное значение (*outgoingNotificationParams*) типа **BSPUnregistrationEvent-NotificationParams** (см. 16.60.2), в

котором компоненты установлены из компонентов *incomingRequestParams* с такими же именами;

h) для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1) создать и отправить сообщение уведомления ПМО БиоАПИ **bspUnregistrationEvent** (см. 13.4) с ИИР главной конечной точки, установленной из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра, установленным *outgoingNotificationParams*;

i) возвратить значение 0 локальному приложению.

16.60.3.2 Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **unregisterBSP** согласно разделу 27, выполняя действия, указанные в 16.60.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.60.3.3 Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_ENDPOINT** локальному приложению.

16.60.4 Когда структура принимает сообщение запроса ПМО БиоАПИ (см. 13.9) **unregisterBSP** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **UnregisterBSP-RequestParams** (см.16.60.2) сообщения запроса ПМО БиоАПИ **unregisterBSP**;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnregisterBSP**, в котором параметры вызова функции установлены путем преобразований из *incomingRequestParams* согласно 16.60.6;

c) в случае, если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ

unregisterBSP (см. 13.3) с возвращаемым значением, установленным на такое значение без выполнения следующих действий;

d) проверить таблицу **VisibleBSPRegistrations** (см. 18.3) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки и

2) компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** *incomingRequestParams*;

e) в случае, если такое поле обнаружено, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unregisterBSP** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_NO_SUCH_COMPONENT_FOUND** без выполнения следующих действий;

f) удалить поле таблицы **VisibleBSPRegistrations**, выполняя действия, указанные в 18.3.3;

g) для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1), создать и отправить сообщение уведомления ПМО БиоАПИ **bspUnregistrationEvent** (см. 13.4) с ИИР главной конечной точки, установленным из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра, установленным на *incomingRequestParams*;

h) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unregisterBSP** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращаемым значением, установленным на 0.

16.60.5 Когда структура принимает сообщение уведомления ПМО БиоАПИ (см. 13.9) **bspUnregistrationEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingNotificationParams* выступать в качестве значения параметра типа **BSPUnregistrationEvent-NotificationParams** (см.

16.60.2) сообщения уведомления ПМО БиоАПИ **bspUnregistrationEvent**;

b) проверить таблицу **VisibleBSPRegistrations** (см. 18.3) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР второстепенной конечной точки и

2) компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** *incomingNotificationParams*;

c) Если такое поле обнаружено, удалить поле таблицы **VisibleBSPRegistrations**, выполняя действия, указанные в 18.3.3;

d) удостовериться, что сообщение подтверждения ПМО БиоАПИ для отправления отсутствует.

16.60.6 Преобразование между параметрами функции Си **BioAPI_UnregisterBSP** и типом ACH.1 **UnregisterBSP-RequestParams** (см. 16.60.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами ACH.1 согласно таблице 119.

Таблица 119 – Преобразование данных между параметрами функции **BioAPI_UnregisterBSP** и типом ACH.1 **UnregisterBSP-RequestParams**

Параметр функции	Компонент типа ACH.1	Подраздел, пункт настоящего стандарта
HostingEndpointIRI	Отсутствует	16.60.7
BSPProductUuid	bspProductUuid	15.58

16.60.7 При преобразовании из параметров функции Си в тип ACH.1, параметр **HostingEndpointIRI** должен быть проигнорирован, так как он уже использовался для определения главной конечной точки. При преобразовании из типа ACH.1 в параметры функции Си, параметр **HostingEndpointIRI** должны быть установлены на **NULL** для определения локальной конечной точки.

16.61 Функция BioAPI_RegisterBFP

16.61.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI_RegisterBFP
        (const uint8_t *HostingEndpointIRI,
        const BioAPI_BFP_SCHEMA *BFPSchema,
        BioAPI_BOOL Update);
```

16.61.2 С данной функцией связаны три типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **registerBFP**, тип сообщения ответа **registerBFP** и тип сообщения уведомления ПМО БиоАПИ **bfpRegistrationEvent**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
RegisterBFP-RequestParams ::= SEQUENCE {
                bfpSchema     BioAPI-BFP-SCHEMA,
                update BOOLEAN
        }
RegisterBFP-ResponseParams ::= NULL
```

и

```
BFPRegistrationEvent-NotificationParams ::= SEQUENCE {
                bfpSchema     BioAPI-BFP-SCHEMA,
                update BOOLEAN
        }
```

16.61.3 Когда структура получает вызов к функции **BioAPI_RegisterBFP** от локального приложения, она сначала определяет ИИР главной конечной точки путем преобразования из параметра **HostingEndpointIRI** согласно 15.3, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.61.3.1 Если главной конечной точкой является локальная конечная точка, структура должна выполнить следующие действия в указанном порядке:

a) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как и во входящем вызове;

b) в случае, если возвращенное значение внутреннего вызова не равно 0, возвратить такое значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*incomingRequestParams*) типа **RegisterBFPRequestParams** (см. 16.61.2) путем преобразования из параметров вызова функции **BioAPI_RegisterBFP** согласно 16.61.6;

d) проверить таблицу **VisibleBFPRegistrations** (см. 18.4) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки; и

2) компонент **bfpProductUuid** имеет такое же значение, как и компонент **bfpProductUuid** компонента **bfpSchema** *incomingRequestParams*;

e) в случае, если такое поле обнаружено и компонент **update** *incomingRequestParams* имеет значение **FALSE**, возвратить значение **BioAPIERR_COMPONENT_ALREADY_REGISTERED** локальному приложению без выполнения следующих действий;

f) в случае, если такое поле обнаружено и компонент **update** *incomingRequestParams* имеет значение **TRUE**, удалить поле таблицы **VisibleBFPRegistrations**, выполняя действия, указанные в 18.4.3;

g) добавить поле в таблицу **VisibleBFPRegistrations** (см. 18.4) , в котором:

1) компонент **hostingEndpointIRI** установлен на ИИР локальной конечной точки; и

2) оставшиеся компоненты установлены из компонентов компонента **bfpSchema** *incomingRequestParams* с такими же именами;

h) создать временное абстрактное значение (*outgoingNotificationParams*) типа **BFPRegistrationEvent- NotificationParams** (см. 16.61.2), в котором все компоненты установлены из компонентов *incomingRequestParams* с такими же именами;

i) для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1) создать и отправить сообщение уведомления ПМО БиоАПИ **bfpRegistrationEvent** (см. 13.4) с ИИР главной конечной точки, установленным из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра, установленным на *outgoingNotificationParams*;

j) возвратить значение 0 локальному приложению.

16.61.3.2 Если главной конечной точкой является второстепенная конечная точка структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **registerBFP** согласно разделу 27, выполняя действия, указанные в 16.61.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.61.3.3 Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_ENDPOINT** локальному приложению.

16.61.4 Когда структура принимает сообщение запроса ПМО БиоАПИ (см. 13.9) **registerBFP** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **RegisterBFP-RequestParams** (см. 16.61.2) сообщения запроса ПМО БиоАПИ **registerBFP**;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_RegisterBFP**, в котором параметры вызова функции должны быть установлены путем преобразований из *incomingRequestParams* согласно 16.61.6;

c) Если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **registerBFP** (см. 13.3) с возвращаемым значением, установленным на это значение без выполнения следующих действий;

d) проверить таблицу **VisibleBFPRegistrations** (см. 18.4) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки и

2) компонент **bfpProductUuid** имеет такое же значение, как и компонент **bfpProductUuid** компонента **bfpSchema** *incomingRequestParams*;

e) Если такое поле обнаружено и компонент **update** *incomingRequestParams* имеет значение **FALSE**, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **registerBFP** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_NO_SUCH_COMPONENT_FOUND** без выполнения следующих действий;

f) Если такое поле обнаружено и компонент **update** *incomingRequestParams* имеет значение **TRUE**, удалить поле таблицы **VisibleBFPRegistrations**, выполняя действия, указанные в 18.4.3;

g) добавить поле в таблицу **VisibleBFPRegistrations** (см. 18.4), в котором:

1) компонент **hostingEndpointIRI** установлен на ИИР локальной конечной точки; и

2) оставшиеся компоненты установлены из компонентов компонента **bfpSchema** *incomingRequestParams* с такими же именами;

h) для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1) создать и отправить сообщение уведомления ПМО БиоАПИ **bfpRegistrationEvent** (см. 13.4) с ИИР главной конечной точки, установленным из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра, установленным на *incomingRequestParams*;

i) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **registerBFP** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращаемым значением, установленным на 0.

16.61.5 Когда структура принимает сообщение уведомления ПМО БиоАПИ (см. 13.9) **bfpRegistrationEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingNotificationParams* выступать в качестве значения параметра типа **BFPRegistrationEvent-NotificationParams** (см. 16.61.2) сообщения уведомления ПМО БиоАПИ **bfpRegistrationEvent**;

b) проверить таблицу **VisibleBFPRegistrations** (см. 18.4) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР второстепенной конечной точки; и

2) компонент **bfpProductUuid** имеет такое же значение, как и компонент **bfpProductUuid** компонента **bfpSchema** *incomingNotificationParams*;

c) в случае, если такое поле обнаружено и компонент **update** *incomingNotificationParams* имеет значение **FALSE**, не выполнять следующие действия;

d) в случае, если такое поле обнаружено и компонент **update** *incomingNotificationParams* имеет значение **TRUE**, удалить поле таблицы **VisibleBFPRegistrations**, выполняя действия, указанные в 18.4.3;

e) добавить поле в таблицу **VisibleBFPRegistrations** (см. 18.4), в котором все компоненты установлены из компонентов компонента **bfpSchema** *incomingNotificationParams* с такими же именами;

f) удостовериться, что сообщение подтверждения ПМО БиоАПИ для отправления отсутствует.

16.61.6 Преобразование между параметрами функции Си **BioAPI_RegisterBFP** и типом АСН.1 **RegisterBFP-RequestParams** (см.

16.61.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 120.

Таблица 120 – Преобразование данных между параметрами функции **BioAPI_RegisterBFP** и типом АСН.1 **UnregisterBSP-RequestParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
HostingEndpointIRI	Отсутствует	16.61.7
BFPSchema	bfpSchema	Раздел 19 совместно с 15.5
Update	update	15.18

16.61.7 При преобразовании из параметров функции Си в тип АСН.1, параметр **HostingEndpointIRI** должен быть проигнорирован, так как он уже использовался для определения главной конечной точки. При преобразовании из типа АСН.1 в параметры функции Си, параметр **HostingEndpointIRI** должен быть установлен на **NULL** для определения локальной конечной точки.

16.62 Функция BioAPI_UnregisterBFP

16.62.1 Данная функция определена в БиоАПИ следующим образом:

```
BioAPI_RETURN BioAPI_UnregisterBFP
        (const uint8_t *HostingEndpointIRI,
        const BioAPI_UUID *BFPProductUuid);
```

16.62.2 С данной функцией связаны три типа сообщений ПМО БиоАПИ: тип сообщения запроса ПМО БиоАПИ **unregisterBFP**, тип сообщения ответа **unregisterBFP**, и тип сообщения уведомления ПМО БиоАПИ **bfpUnregistrationEvent**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
UnregisterBFP-RequestParams ::= SEQUENCE {
                bfpProductUuid              BioAPI-UUID
        }
```

```
UnregisterBFP-ResponseParams ::= NULL
```

и

```
BFPUnregistrationEvent-NotificationParams ::= SEQUENCE {
        bfpProductUuid          BioAPI-UUID
    }
```

16.62.3 Когда структура получает вызов к функции **BioAPI_UnregisterBFP** от локального приложения, она сначала определяет ИИР главной конечной точки путем преобразования из параметра **HostingEndpointIRI** согласно 15.3, а затем выполняют действия, указанные в одном из следующих подпунктов.

16.62.3.1 Если главной конечной точкой является локальная конечная точка, структура должна выполнить следующие действия в указанном порядке:

a) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к такой же функции с теми же значениями параметра, как и во входящем вызове;

b) в случае, если возвращенное значение внутреннего вызова не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

c) создать временное абстрактное значение (*incomingRequestParams*) типа **UnregisterBFPRequestParams** (см. 16.62.2) путем преобразования из параметров вызова функции **BioAPI_UnregisterBFP** согласно 16.62.6;

d) проверить таблицу **VisibleBFPRegistrations** (см. 18.4) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки; и

2) компонент **bfpProductUuid** имеет такое же значение, как и компонент **bfpProductUuid** *incomingRequestParams*;

e) в случае, если такое поле не обнаружено, возвратить значение **BioAPIERR_NO_SUCH_COMPONENT_FOUND** локальному приложению без выполнения следующих действий;

f) удалить поле таблицы **VisibleBFPRegistrations**, выполняя действия, указанные в 18.4.3;

g) создать временное абстрактное значение (*outgoingNotificationParams*) типа **BFPUnregistrationEvent-NotificationParams** (см. 16.62.2), в котором все компоненты должны быть установлены из компонентов *incomingRequestParams* с такими же именами;

h) для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1) создать и отправить сообщение уведомления ПМО БиоАПИ **bfpUnregistrationEvent** (см. 13.4) с ИИР главной конечной точки, установленным из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра, установленным на *outgoingNotificationParams*;

i) возвратить значение 0 локальному приложению.

16.62.3.2 Если главной конечной точкой является второстепенная конечная точкая структуры, структура должна обработать вызов путем обмена с главной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ **unregisterBFP** согласно разделу 27, выполняя действия, указанные в 16.62.6, для преобразования между параметрами функции и компонентами АСН.1, если это установлено в указанном разделе.

16.62.3.3 Если главная конечная точка не может быть определена, структура должна возвратить значение **BioAPIERR_UNABLE_TO_LOCATE_ENDPOINT** локальному приложению.

16.62.4 Когда структура принимает сообщение запроса ПМО БиоАПИ (см. 13.9) **unregisterBFP** от главной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingRequestParams* выступать в качестве значения параметра типа **UnregisterBFP-RequestParams** (см. 16.62.2) сообщения запроса ПМО БиоАПИ **unregisterBFP**;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnregisterBFP**, в котором параметры вызова функции должны

быть установлены путем преобразований из *incomingRequestParams* согласно 16.62.6;

c) случае, если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unregisterBFP** (см. 13.3) с возвращаемым значением, установленным на это значение без выполнения следующих действий;

d) проверить таблицу **VisibleBFPRegistrations** (см. 18.4) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР локальной конечной точки и

2) компонент **bfpProductUuid** имеет такое же значение, как и компонент **bfpProductUuid** *incomingRequestParams*;

e) Если такое поле обнаружено, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unregisterBFP** (см. 13.3) с возвращаемым значением, установленным на **BioAPIERR_NO_SUCH_COMPONENT_FOUND** без выполнения следующих действий;

f) удалить поле таблицы **VisibleBFPRegistrations**, выполняя действия, указанные в 18.4.3;

g) для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1) создать и отправить сообщение уведомления ПМО БиоАПИ **bfpUnregistrationEvent** (см. 13.4) с ИИР главной конечной точки, установленным из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра, установленным *incomingRequestParams*;

h) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ **unregisterBFP** (см. 13.3) со значением параметра, установленным на **NULL**, и возвращаемым значением, установленным на 0.

16.62.5 Когда структура принимает сообщение уведомления ПМО БиоАПИ (см. 13.9) **bfpUnregistrationEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingNotificationParams* выступать в качестве значения параметра типа **BFPUnregistrationEvent-NotificationParams** (см. 16.62.2) сообщения уведомления ПМО БиоАПИ **bfpUnregistrationEvent**;

b) проверить таблицу **VisibleBFPRegistrations** (см. 18.4) на наличие поля, в котором:

1) компонент **hostingEndpointIRI** содержит ИИР второстепенной конечной точки и

2) компонент **bfpProductUuid** имеет такое же значение, как и компонент **bfpProductUuid** *incomingNotificationParams*;

c) Если такое поле обнаружено, удалить поле таблицы **VisibleBFPRegistrations** (см. 18.4.3);

d) удостовериться, что сообщение подтверждения ПМО БиоАПИ для отправления отсутствует.

16.62.6 Преобразование между параметрами функции Си **BioAPI_UnregisterBFP** и типом АСН.1 **UnregisterBFP-RequestParams** (см. 16.62.2) выполняют путем преобразования между индивидуальными параметрами функции и компонентами АСН.1 согласно таблице 121.

Таблица 121 – Преобразование данных между параметрами функции **BioAPI_UnregisterBFP** и типом АСН.1 **UnregisterBFP-RequestParams**

Параметр функции	Компонент типа АСН.1	Подраздел, пункт настоящего стандарта
HostingEndpointIRI	Отсутствует	16.62.7
BFPProductUuid	bfpProductUuid	15.58

16.62.7 При преобразовании из параметров функции Си в тип АСН.1, параметр **HostingEndpointIRI** должен быть проигнорирован, так как он уже использовался для определения главной конечной точки. При преобразовании из типа АСН.1 в параметры функции Си, параметр **HostingEndpointIRI** должен быть установлен на **NULL** для определения локальной конечной точки.

17 Функции обратного вызова, определенные в БиоАПИ, и соответствующие сообщения ПМО БиоАПИ

17.1 Функция обратного вызова BioAPI_EVENT_HANDLER

17.1.1 В БиоАПИ тип указателя функции Си данной функции обратного вызова определен следующим образом:

```
typedef BioAPI_RETURN (*BioAPI_EVENT_HANDLER)
        (const BioAPI_UUID *BSPUuid,
        BioAPI_UNIT_ID UnitID,
        void *EventHandlerCtx,
        const BioAPI_UNIT_SCHEMA *UnitSchema,
        BioAPI_EVENT EventType);
```

17.1.2 В БиоАПИ соответствующий тип указателя функции Си в интерфейсе БиоППИ определен следующим образом:

```
typedef BioAPI_RETURN (*BioSPI_EVENT_HANDLER)
        (const BioAPI_UUID *BSPUuid,
        BioAPI_UNIT_ID UnitID,
        const BioAPI_UNIT_SCHEMA *UnitSchema,
        BioAPI_EVENT EventType);
```

17.1.3 С данной функцией связан один тип сообщений ПМО БиоАПИ: тип сообщения уведомления **unitEvent**, который переносит значение следующего параметра типа АСН.1 сообщений ПМО БиоАПИ:

```
UnitEvent-NotificationParams ::= SEQUENCE {
                bspProductUuid      BioAPI-UUID,
                unitID              BioAPI-UNIT-ID,
                unitSchema          BioAPI-UNIT-SCHEMA OPTIONAL,
                unitEventType       BioAPI-UNIT-EVENT-TYPE
        }
```

17.1.4 Следующие типы АСН.1 применяют при спецификации поведения структуры, но их абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
UnitEventHandlerCallbackParams ::= SEQUENCE {
        unitEventHandlerAddress     MemoryAddress,
        unitEventHandlerContext     MemoryAddress,
        bspUuid                     BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        unitSchema                  BioAPI-UNIT-SCHEMA OPTIONAL,
        unitEventType                   BioAPI-UNIT-EVENT-TYPE
    }
UnitEventInfo ::= SEQUENCE {
        hostingEndpointIRI          EndpointIRI,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        unitSchema                  BioAPI-UNIT-SCHEMA OPTIONAL,
        unitEventType               BioAPI-UNIT-EVENT-TYPE
    }
```

17.1.5 Когда структура получает вызов к функции обратного вызова **BioSPI_EVENT_HANDLER** от ПБУ, она должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*incomingNotificationParams*) типа **UnitEvent-NotificationParams** (см. 17.1.3) путем преобразования из параметров вызова функции согласно 17.1.7;

b) создать временное абстрактное значение (*eventInfo*) типа **UnitEventInfo** (см. 17.1.4), в котором компонент **hostingEndpointIRI** должны быть установлены на ИИР локальной конечной точки, а оставшиеся компоненты должны быть установлены из компонентов *incomingNotificationParams* с такими же именами;

c) зарегистрировать модуль операций, основанный на *eventInfo*, на 0 или более подписчиков (обработчик модуля операций локального приложения или главные конечные точки) согласно разделу 29; и

d) возвратить ПБУ значение 0.

17.1.6 Когда структура принимает сообщение уведомления ПМО БиоАПИ (см. 13.9) **unitEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingNotificationParams* выступать в качестве значения параметра типа **UnitEvent-NotificationParams** (см. 17.1.3) сообщения уведомления ПМО БиоАПИ **unitEvent**;

b) создать временное абстрактное значение (*eventInfo*) типа **UnitEventInfo** (см. 17.1.4), в котором компонент **hostingEndpointIRI** установлен из компонента **slaveEndpointIRI** из сообщения уведомления ПМО БиоАПИ, а оставшиеся компоненты – из компонентов *incomingNotificationParams* с такими же именами;

c) зарегистрировать модуль операций, основанный на *eventInfo*, на 0 или более подписчиков (обработчик модуля операций локального приложения или главные конечные точки) согласно разделу 29;

d) удостовериться, что сообщение подтверждения ПМО БиоАПИ для отправления отсутствует.

17.1.7 Преобразование из параметров функции Си обратного вызова **BioSPI_EVENT_HANDLER** в тип ACH.1 **UnitEvent-NotificationParams** (см. 17.1.3) выполняют путем преобразования из индивидуальных параметров функции в компоненты ACH.1 согласно таблице 122.

Таблица 122 – Преобразование данных из параметров функции обратного вызова **BioSPI_EVENT_HANDLER** в тип ACH.1 **UnitEvent-NotificationParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
SPUuid	bspProductUuid	Раздел 19 совместно с 15.58
UnitID	unitID	15.55
UnitSchema	unitSchema	Раздел 19 совместно с 15.57
EventType	unitEventType	15.30

17.1.8 Преобразование из типа АСН.1 **UnitEventHandlerCallbackParams** (см. 17.1.4) в параметры функции Си обратного вызова **BioAPI_EVENT_HANDLER** выполняют путем преобразования из индивидуальных компонентов АСН.1 в параметры функции согласно таблице 123.

Таблица 123 – Преобразование данных из типа АСН.1 **UnitEventHandlerCallbackParams** в параметры функции Си обратного вызова **BioAPI_EVENT_HANDLER**

Компонент типа АСН.1	Параметр функции	Раздел, пункт настоящего стандарта
unitEventHandlerAddress	Отсутствует	17.1.9
unitEventHandlerContext	EventHandlerCtx	15.1.7
bspUuid	BSPUuid	Раздел 19 совместно с 15.58
unitID	UnitID	15.55
unitSchema	UnitSchema	Раздел 19 совместно с 15.57
unitEventType	EventType	15.30

17.1.9 Компонент **unitEventHandlerAddress** соответствует адресу обратного вызова (а не параметру) функции обратного вызова. Преобразование выполняют, применяя действия, указанные в 15.1.7.

17.2 Функция обратного вызова BioAPI_GUI_SELECT_EVENT_HANDLER

17.2.1 В БиоАПИ тип указателя функции Си данной функции обратного вызова определен следующим образом:

```
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,
    void *GUISelectEventHandlerCtx,
    BioAPI_GUI_OPERATION Operation,
    BioAPI_GUI_MOMENT Moment,
    BioAPI_RETURN ResultCode,
    uint32_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);
```

17.2.2 В БиоАПИ соответствующий тип указателя функции Си в интерфейсе БиоППИ определен следующим образом:

```
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,
    uint32_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);
```

17.2.3 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения уведомления ПМО БиоАПИ **guiSelectEvent** и тип сообщения подтверждения **guiSelectEvent**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
GUISelectEvent-NotificationParams ::= SEQUENCE {
        guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-HANDLE OPTIONAL,
        enrollType                  BioAPI-GUI-ENROLL-TYPE,
        operation                   BioAPI-GUI-OPERATION,
        moment                      BioAPI-GUI-MOMENT,
        resultCode                  BioAPI-RETURN,
```

```
                maxNumEnrollSamples         UnsignedInt,
                selectableInstances         BioAPI-BIR-SUBTYPE-MASK,
                capturedInstances           BioAPI-BIR-SUBTYPE-MASK,
                text UTF8String                 OPTIONAL
        }
```

и

```
GUISelectEvent-AcknowledgementParams ::= SEQUENCE {
                selectedInstances           BioAPI-BIR-SUBTYPE-MASK,
                response                    BioAPI-GUI-RESPONSE
        }
```

17.2.4 Следующие типы АСН.1 применяют при спецификации поведения структуры, но их абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
GUISelectEventHandlerCallbackParams ::= SEQUENCE {
                guiSelectEventHandlerAddress    MemoryAddress,
                guiSelectEventHandlerContext    MemoryAddress,
                bspUuid                         BioAPI-UUID,
                unitID                          BioAPI-UNIT-ID,
                bspHandle                       BioAPI-HANDLE OPTIONAL,
                enrollType                      BioAPI-GUI-ENROLL-TYPE,
                operation                       BioAPI-GUI-OPERATION,
                moment                          BioAPI-GUI-MOMENT,
                resultCode                      BioAPI-RETURN,
                maxNumEnrollSamples             UnsignedInt,
                selectableInstances             BioAPI-BIR-SUBTYPE-MASK,
                capturedInstances               BioAPI-BIR-SUBTYPE-MASK,
                text UTF8String                 OPTIONAL
        }
GUISelectEventInfo ::= SEQUENCE {
                subscriberEndpointIRI           EndpointIRI,
                guiEventSubscriptionUuid        BioAPI-UUID OPTIONAL,
                hostingEndpointIRI              EndpointIRI,
                bspProductUuid                  BioAPI-UUID,
                unitID                          BioAPI-UNIT-ID,
                originalBSPHandle               BioAPI-HANDLE OPTIONAL,
                enrollType                      BioAPI-GUI-ENROLL-TYPE,
                operation                       BioAPI-GUI-OPERATION,
                moment                          BioAPI-GUI-MOMENT,
                resultCode                      BioAPI-RETURN,
                maxNumEnrollSamples             UnsignedInt,
```

selectableInstances	**BioAPI-BIR-SUBTYPE-MASK,**
capturedInstances	**BioAPI-BIR-SUBTYPE-MASK,**
text UTF8String	**OPTIONAL**

}

17.2.5 Когда структура получает вызов к функции обратного вызова **BioSPI_GUI_SELECT_EVENT_HANDLER** от ПБУ, она должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*incomingNotificationParams*) типа **GUISelectEvent-NotificationParams** (см. 17.2.3) путем преобразования из параметров вызова функции согласно 17.2.7;

b) проверить таблицу **GUIEventRedirectors** (см. 18.12) на наличие поля, в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** из *incomingNotificationParams*, а компонент **GUISelectEventRedirected** имеет значение **TRUE**;

c) Если соответствующее поле отсутствует, создать временное абстрактное значение (*eventInfo*) типа **GUISelectEventInfo** (см. 17.2.4), в котором:

1) в случае, если таблица **AttachSessionRemoteReferences** (см. 18.9) имеет поле, в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** из *incomingNotificationParams*, компонент **subscriberEndpointIRI** из *eventInfo* должен быть установлен из компонента **referrerEndpointIRI** этого поля; в противном случае, он должен быть установлен на ИИР локальной конечной точки;

2) необязательный компонент **guiEventSubscriptionUuid** должен отсутствовать;

3) компонент **hostingEndpointIRI** должен быть установлен на ИИР локальной конечной точки; и

4) оставшиеся компоненты должны быть установлены из компонентов *incomingNotificationParams* с такими же именами;

d) Если существует одно соответствующее поле (*redirector*), создать временное абстрактное значение (*eventInfo*) типа **GUISelectEventInfo** (см. 17.2.4), в котором:

1) компонент **subscriberEndpointIRI** должен быть установлен из компонента **subscriberEndpointIRI** *redirector*;

2) необязательный компонент **guiEventSubscriptionUuid** должен присутствовать и быть установлен из компонента **guiEventSubscriptionUuid** из *redirector*;

3) компонент **hostingEndpointIRI** установлен на ИИР локальной конечной точки;

4) необязательный компонент **originalBSPHandle** должен отсутствовать и

5) оставшиеся компоненты установлены из компонентов *incomingNotificationParams* с такими же именами;

e) зарегистрировать операцию выбора ГИП, основанную на *eventInfo*, на подписчика (либо обработчика операции выбора ГИП локального приложения, либо главная конечная точка), и определить значение параметра подтверждения (читай *incomingAcknowledgementParams*), а также определить возвращаемое значение подтверждения (*incomingReturnValue*) согласно разделу 30;

f) в случае, если *incomingReturnValue* не равно 0, возвратить это значение ПБУ без выполнения нижеследующих действий;

g) установить исходящие параметры вызова к функции обратного вызова **BioSPI_GUI_SELECT_EVENT_HANDLER** путем преобразования из *incomingAcknowledgementParams* согласно 17.2.9; и

h) возвратить значение 0 ПБУ.

17.2.6 Когда структура принимает сообщение уведомления ПМО БиоАПИ (см. 13.9) **guiSelectEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingNotificationParams* выступать в качестве значения параметра типа **GUISelectEvent- NotificationParams** (см. 17.2.3) сообщения уведомления ПМО БиоАПИ;

b) создать временное абстрактное значение (*eventInfo*) типа **GUISelectEventInfo** (см. 17.2.4), в котором:

1) компоненты **subscriberEndpointIRI** установлены на ИИР локальной конечной точки;

2) компонент **hostingEndpointIRI** установлены из компонента **slaveEndpointIRI** сообщения уведомления ПМО БиоАПИ и

3) оставшиеся компоненты установлены из компонентов *incomingNotificationParams* с такими же именами;

c) зарегистрировать операцию выбора ГИП, основанную на *eventInfo*, на подписчика (либо обработчика операции выбора ГИП локального приложения либо главная конечная точка) и определить значение параметра подтверждения (*incomingAcknowledgementParams*), а также определить возвращаемое значение подтверждения (*incomingReturnValue*) согласно разделу 30;

d) Если *incomingReturnValue* не равно 0, создать и отправить соответствующее сообщение подтверждения ПМО БиоАПИ **guiSelectEvent** (см. 13.5) с возвращаемым значением, установленным на это значение, без выполнения нижеследующих действий;

e) создать и отправить соответствующее сообщение подтверждения ПМО БиоАПИ **guiSelectEvent** (см. 13.5) со значением параметра, установленным на *incomingAcknowledgementParams*, и с возвращаемым значением, установленным на 0.

17.2.7 Преобразование из параметров функции Си обратного вызова **BioSPI_GUI_SELECT_EVENT_HANDLER** в тип АСН.1 **GUISelectEvent-NotificationParams** (см. 17.2.3) выполняют путем преобразования из

индивидуальных параметров функции в компоненты ACH.1 согласно таблице 124.

Таблица 124 – Преобразование данных из параметров функции обратного вызова **BioSPI_GUI_SELECT_EVENT_HANDLER** в тип ACH.1 **GUISelectEvent-NotificationParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
Отсутствует	guiEventSubscriptionUuid	17.2.8
BSPUuid	bspProductUuid	Раздел 19 совместно с 15.58
UnitID	unitID	15.55
BSPHandle	originalBSPHandle	Раздел 19 совместно с 15.42
EnrollType	enrollType	15.38
Operation	operation	15.39
Moment	moment	15.37
ResultCode	resultCode	15.52
MaxNumEnrollSamples	maxNumEnrollSamples	15.1.6
SelectableInstances	selectableInstances	15.17
CapturedInstances	capturedInstances	15.17
Text	text	15.2
SelectedInstances	Отсутствует	Раздел 22
Response	Отсутствует	Раздел 22

17.2.8 Необязательный компонент **guiEventSubscriptionUuid** должен отсутствовать.

17.2.9 Преобразование из типа ACH.1 **GUISelectEvent-AcknowledgementParams** (см. 17.2.3) в параметры функции Си обратного

вызова **BioSPI_GUI_SELECT_EVENT_HANDLER** выполняют путем преобразования из индивидуальных компонентов АСН.1 в параметры функции согласно таблице 125.

Таблица 125 – Преобразование данных из типа АСН.1 **GUISelectEvent-AcknowledgementParams** в параметры функции Си обратного вызова **BioSPI_GUI_SELECT_EVENT_HANDLER**

Компонент типа АСН.1	Параметр функции	Раздел, пункт настоящего стандарта
selectedInstances	SelectedInstances	Раздел 20 совместно с 15.17
response	Response	Раздел 20 совместно с 15.40

17.2.10 Преобразование из типа АСН.1 **GUISelectEventHandlerCallbackParams** (см. 17.2.4) в параметры функции Си обратного вызова **BioAPI_GUI_SELECT_EVENT_HANDLER** выполняют путем преобразования из индивидуальных компонентов АСН.1 в параметры функции согласно таблице 126.

Таблица 126 – Преобразования данных из типа АСН.1 **GUISelectEventHandlerCallbackParams** в параметры функции Си обратного вызова **BioAPI_GUI_SELECT_EVENT_HANDLER**

Компонент типа АСН.1	Параметр функции	Раздел, пункт настоящего стандарта
guiSelectEventHandlerAddress	Отсутствует	17.2.11
guiSelectEventHandlerContext	GUISelectEventHandlerCtx	15.1.7
bspUuid	BSPUuid	Раздел 19 совместно с 15.58
unitID	UnitID	15.55
bspHandle	BSPHandle	Раздел 19 совместно с 15.42
operation	Operation	15.39

Окончание таблицы 126

Компонент типа ACH.1	Параметр функции	Раздел, пункт настоящего стандарта
moment	Moment	15.37
resultCode	ResultCode	15.52
maxNumEnrollSamples	MaxNumEnrollSamples	15.1.6
selectableInstances	SelectableInstances	15.17
capturedInstances	CapturedInstances	15.17
text	Text	15.2
Отсутствует	SelectedInstances	Раздел 22
Отсутствует	Response	Раздел 22

17.2.11 Компонент **guiSelectEventHandlerAddress** соответствует адресу обратного вызова, а не параметру функции обратного вызова. Преобразование выполняют применяя действия, указанные в 15.1.7.

17.2.12 Преобразование из параметров функции Си обратного вызова **BioAPI_GUI_SELECT_EVENT_HANDLER** в тип ACH.1 **GUISelectEvent-AcknowledgementParams** (см. 17.2.3) выполняют путем преобразования из индивидуальных параметров функции в компоненты ACH.1 согласно таблице 127.

Таблица 127 – Преобразование данных из параметров функции обратного вызова **BioAPI_GUI_SELECT_EVENT_HANDLER** в тип ACH.1 **GUISelectEvent-AcknowledgementParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
SelectedInstances	selectedInstances	Раздел 20 совместно с 15.17
Response	response	Раздел 20 совместно с 15.40

17.3 Функция обратного вызова BioAPI_GUI_STATE_EVENT_HANDLER

17.3.1 В БиоАПИ тип указателя функции Си данной функции обратного вызова определен следующим образом:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_STATE_EVENT_HANDLER)
    (const BioAPI_UUID *BSPUuid,
    BioAPI_UNIT_ID UnitID,
    const BioAPI_HANDLE *BSPHandle,
    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);
```

17.3.2 В БиоАПИ соответствующий тип указателя функции Си в интерфейсе БиоППИ определен следующим образом:

```
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);
```

17.3.3 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения уведомления ПМО БиоАПИ **guiStateEvent** и тип сообщения

подтверждения **guiStateEvent**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
GUIStateEvent-NotifcationParams ::= SEQUENCE {
        guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        originalBSPHandle           BioAPI-HANDLE OPTIONAL,
        operation                   BioAPI-GUI-OPERATION,
        suboperation                BioAPI-GUI-SUBOPERATION,
        purpose                     BioAPI-BIR-PURPOSE,
        moment                      BioAPI-GUI-MOMENT,
        resultCode                  BioAPI-RETURN,
        enrollSampleIndex           SignedInt,
        bitmaps                     BioAPI-GUI-BITMAP-ARRAY OPTIONAL,
        text UTF8String             OPTIONAL
    }
```

и

```
GUIStateEvent-AcknowledgementParams ::= SEQUENCE {
        response                        BioAPI-GUI-RESPONSE,
        enrollSampleIndexToRecapture    SignedInt
    }
```

17.3.4 Следующие типы АСН.1 применяют при спецификации поведения структуры, но их абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
GUIStateEventHandlerCallbackParams ::= SEQUENCE {
        guiStateEventHandlerAddress     MemoryAddress,
        guiStateEventHandlerContext     MemoryAddress,
        bspUuid                         BioAPI-UUID,
        unitID                          BioAPI-UNIT-ID,
        bspHandle                       BioAPI-HANDLE OPTIONAL,
        operation                       BioAPI-GUI-OPERATION,
        suboperation                    BioAPI-GUI-SUBOPERATION,
        purpose                         BioAPI-BIR-PURPOSE,
        moment                          BioAPI-GUI-MOMENT,
        resultCode                      BioAPI-RETURN,
        enrollSampleIndex               SignedInt,
        bitmaps                         BioAPI-GUI-BITMAP-ARRAY OPTIONAL,
        text UTF8String                 OPTIONAL
    }
```

```
GUIStateEventInfo ::= SEQUENCE {
        subscriberEndpointIRI          EndpointIRI,
        guiEventSubscriptionUuid       BioAPI-UUID OPTIONAL,
        hostingEndpointIRI             EndpointIRI,
        bspProductUuid                 BioAPI-UUID,
        unitID                         BioAPI-UNIT-ID,
        originalBSPHandle              BioAPI-HANDLE OPTIONAL,
        operation                      BioAPI-GUI-OPERATION,
        suboperation                   BioAPI-GUI-SUBOPERATION,
        purpose                        BioAPI-BIR-PURPOSE,
        moment                         BioAPI-GUI-MOMENT,
        resultCode                     BioAPI-RETURN,
        enrollSampleIndex              SignedInt,
        bitmaps                        BioAPI-GUI-BITMAP-ARRAY OPTIONAL,
        text UTF8String                OPTIONAL
    }
```

17.3.5 Когда структура получает вызов к функции обратного вызова **BioSPI_GUI_STATE_EVENT_HANDLER** от ПБУ, она должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*incomingNotificationParams*) типа **GUIStateEvent-NotifcationParams** (см. 17.3.3) путем преобразования из параметров вызова функции согласно 17.3.7;

b) проверить таблицу **GUIEventRedirectors** (см. 18.12) на наличие поля, в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** из *incomingNotificationParams*, а компонент **guiStateEventRedirected** имеет значение **TRUE**;

c) Если соответствующее поле отсутствует, создать временное абстрактное значение (читай *eventInfo*) типа **GUIStateEventInfo** (см. 17.3.4), в котором:

1) в случае, если таблица **AttachSessionRemoteReferences** (см. 18.9) имеет поле, в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** из *incomingNotificationParams*, компонент **subscriberEndpointIRI** из

eventInfo должен быть установлен из компонента **referrerEndpointIRI** такого поля; в противном случае – на ИИР локальной конечной точки;

2) необязательный компонент **guiEventSubscriptionUuid** должен отсутствовать;

3) компонент **hostingEndpointIRI** должен быть установлен на ИИР локальной конечной точки; и

4) оставшиеся компоненты должны быть установлены из компонентов *incomingNotificationParams* с такими же именами;

d) в случае, если существует одно соответствующее поле (*redirector*), создать временное абстрактное значение (*eventInfo*) типа **GUIStateEventInfo** (см. 17.3.4), в котором:

1) компонент **subscriberEndpointIRI** должен быть установлен из компонента **subscriberEndpointIRI** *redirector*;

2) необязательный компонент **guiEventSubscriptionUuid** должен присутствовать и быть установлен из компонента **guiEventSubscriptionUuid** *redirector*;

3) компонент **hostingEndpointIRI** должен быть установлен на ИИР локальной конечной точки;

4) необязательный компонент **originalBSPHandle** должен отсутствовать и

5) оставшиеся компоненты должны быть установлены из компонентов *incomingNotificationParams* с такими же именами;

e) зарегистрировать операцию состояния ГИП, основанную на *eventInfo*, на подписчика (либо обработчика операций состояния ГИП локального приложения, либо главная конечная точка) и определить значение параметра подтверждения (*incomingAcknowledgementParams*), а также определить возвращаемое значение подтверждения (*incomingReturnValue*) согласно разделу 30;

f) в случае, если *incomingReturnValue* не равно 0, вернуть это значение ПБУ, без выполнения нижеследующих действий;

g) установить исходящие параметры вызова к функции обратного вызова **BioSPI_GUI_STATE_EVENT_HANDLER** путем преобразования из *incomingAcknowledgementParams* согласно 17.3.9; и

h) возвратить ПБУ значение 0.

17.3.6 Когда структура принимает сообщение уведомления ПМО БиоАПИ (см. 13.9) **guiStateEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingNotificationParams* выступать в качестве значения параметра типа **GUIStateEvent-NotifcationParams** (см. 17.3.3) сообщения уведомления ПМО БиоАПИ;

b) создать временное абстрактное значение (*eventInfo*) типа **GUIStateEventInfo** (см. 17.3.4), в котором:

1) компоненты **subscriberEndpointIRI** установлены на ИИР локальной конечной точки;

2) компонент **hostingEndpointIRI** установлен из компонента **slaveEndpointIRI** сообщения уведомления ПМО БиоАПИ и

3) оставшиеся компоненты установлены из компонентов *incomingNotificationParams* с такими же именами;

c) зарегистрировать операцию состояния ГИП, основанную на *eventInfo*, на подписчика (либо обработчика операций состояния ГИП локального приложения, либо главная конечная точка) и определить значение параметра подтверждения (*incomingAcknowledgementParams*), а также определить возвращаемое значение подтверждения (читай *incomingReturnValue*) согласно разделу 31;

d) Если *incomingReturnValue* не равно 0, создать и отправить соответствующее сообщение подтверждения ПМО БиоАПИ

guiStateEvent (см. 13.5) с возвращаемым значением, установленным на это значение, без выполнения следующих действий;

е) создать и отправить соответствующее сообщение подтверждения ПМО БиоАПИ **guiStateEvent** (см. 13.5) со значением параметра, установленным на *incomingAcknowledgementParams*, и с возвращаемым значением, установленным на 0.

17.3.7 Преобразование из параметров функции Си обратного вызова **BioSPI_GUI_STATE_EVENT_HANDLER** в тип АСН.1 **GUIStateEvent-NotificationParams** (см. 17.3.3) выполняют путем преобразования из индивидуальных параметров функции в компоненты АСН.1 согласно таблице 128.

Таблица 128 – Преобразование данных из параметров функции обратного вызова **BioSPI_GUI_STATE_EVENT_HANDLER** в тип ACH.1 **GUIStateEvent-NotifcationParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
Отсутствует	guiEventSubscriptionUuid	17.3.8
BSPUuid	bspProductUuid	Раздел 19 совместно с 15.58
UnitID	unitID	15.55
BSPHandle	originalBSPHandle	Раздел 19 совместно с 15.42
Operation	operation	15.39
Suboperation	suboperation	15.41
Purpose	purpose	15.14
Moment	moment	15.37
ResultCode	resultCode	15.52
EnrollSampleIndex	enrollSampleIndex	15.1.6
Bitmaps	bitmaps	Раздел 19 совместно с 15.35
Text	text	15.2
Response	Отсутствует	Раздел 22
EnrollSampleIndexToRecapture	Отсутствует	Раздел 22

17.3.8 Необязательный компонент **guiEventSubscriptionUuid** должен отсутствовать.

17.3.9 Преобразование из типа ACH.1 **GUIStateEvent-AcknowledgementParams** (см. 17.3.3) в параметры функции Си обратного вызова **BioSPI_GUI_STATE_EVENT_HANDLER** выполняют путем преобразования из индивидуальных компонентов ACH.1 в параметры функции согласно таблице 129.

Таблица 129 – Преобразование данных из типа ACH.1 **GUIStateEvent-AcknowledgementParams** в параметры функции Си обратного вызова **BioSPI_GUI_STATE_EVENT_HANDLER**

Компонент типа ACH.1	Параметр функции	Раздел, пункт настоящего стандарта
response	Response	Раздел 20 совместно с 15.40
enrollSampleIndexToRecapture	EnrollSampleIndexToRecapture	Раздел 20 совместно с 15.1.6

17.3.10 Преобразование из типа ACH.1 **GUIStateEventHandlerCallbackParams** (см. 17.3.4) в параметры функции Си обратного вызова **BioAPI_GUI_STATE_EVENT_HANDLER** выполняют путем преобразования из индивидуальных компонентов ACH.1 в параметры функции согласно таблице 130.

Таблица 130 – Преобразования данных из типа ACH.1 **GUIStateEventHandlerCallbackParams** в параметры функции Си обратного вызова **BioAPI_GUI_STATE_EVENT_HANDLER**

Компонент типа ACH.1	Параметр функции	Раздел, пункт настоящего стандарта
guiStateEventHandlerAddress	Отсутствует	17.3.11
guiStateEventHandlerContext	GUIStateEventHandlerCtx	15.1.7
bspUuid	BSPUuid	Раздел 19 совместно с 15.58
unitID	UnitID	15.55
bspHandle	BSPHandle	Раздел 19 совместно с 15.42
operation	Operation	15.39
suboperation	Suboperation	15.41
purpose	Purpose	15.14
moment	Moment	15.37
resultCode	ResultCode	15.52
enrollSampleIndex	EnrollSampleIndex	15.1.6
bitmaps	Bitmaps	Раздел 19 совместно с 15.35
text	Text	15.2
Отсутствует	Response	Раздел 22
Отсутствует	EnrollSampleIndexToRecapture	Раздел 22

17.3.11 Компонент **guiStateEventHandlerAddress** соответствует адресу обратного вызова, а не параметру функции обратного вызова. Преобразование выполняют применяя действия, указанные в 15.1.7.

17.3.12 Преобразование из параметров функции Си обратного вызова **BioAPI_GUI_STATE_EVENT_HANDLER** в тип ACH.1 **GUIStateEvent-AcknowledgementParams** (см. 17.3.3) выполняют путем преобразования из индивидуальных параметров функции в компоненты ACH.1 согласно таблице 131.

Таблица 131 – Преобразование данных из параметров функции обратного вызова **BioAPI_GUI_STATE_EVENT_HANDLER** в тип ACH.1 **GUIStateEvent-AcknowledgementParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
Response	response	Раздел 20 совместно с 15.40
EnrollSampleIndexToRecapture	enrollSampleIndexToRecapture	Раздел 20 совместно с 15.1.6

17.4 Функция обратного вызова BioAPI_GUI_PROGRESS_EVENT_HANDLER

17.4.1 В БиоАПИ тип указателя функции Си данной функции обратного вызова определен следующим образом:

```
typedef BioAPI_RETURN (BioAPI *BioAPI_GUI_PROGRESS_EVENT_HANDLER)
    (const BioAPI_UUID *BSPUuid,
```

```
    BioAPI_UNIT_ID UnitID,
    const BioAPI_HANDLE *BSPHandle,
    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);
```

17.4.2 В БиоАПИ соответствующий тип указателя функции Си в интерфейсе БиоППИ определен следующим образом:

```
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);
```

17.4.3 С данной функцией связаны два типа сообщений ПМО БиоАПИ: тип сообщения уведомления ПМО БиоАПИ **guiProgressEvent** и тип сообщения подтверждения **guiProgressEvent**, которые переносят значение следующего параметра типов АСН.1 сообщений ПМО БиоАПИ соответственно:

```
GUIProgressEvent-NotificationParams ::= SEQUENCE {
        guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        originalBSPHandle           BioAPI-HANDLE OPTIONAL,
        operation                   BioAPI-GUI-OPERATION,
        suboperation                BioAPI-GUI-SUBOPERATION,
        purpose                     BioAPI-BIR-PURPOSE,
        moment                      BioAPI-GUI-MOMENT,
```

```
            suboperationProgress        UnsignedByte,
            bitmaps                     BioAPI-GUI-BITMAP-ARRAY OPTIONAL,
            text UTF8String             OPTIONAL
        }
```

и

```
    GUIProgressEvent-AcknowledgementParams ::= SEQUENCE {
            response                    BioAPI-GUI-RESPONSE
        }
```

17.4.4 Следующие типы АСН.1 применяют при спецификации поведения структуры, но их абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ:

```
    GUIProgressEventHandlerCallbackParams ::= SEQUENCE {
            guiProgressEventHandlerAddress    MemoryAddress,
            guiProgressEventHandlerContext    MemoryAddress,
            bspUuid                           BioAPI-UUID,
            unitID                            BioAPI-UNIT-ID,
            bspHandle                         BioAPI-HANDLE OPTIONAL,
            operation                         BioAPI-GUI-OPERATION,
            suboperation                      BioAPI-GUI-SUBOPERATION,
            purpose                           BioAPI-BIR-PURPOSE,
            moment                            BioAPI-GUI-MOMENT,
            suboperationProgress              UnsignedByte,
            bitmaps                           BioAPI-GUI-BITMAP-ARRAY OPTIONAL,
            text                              UTF8String OPTIONAL
        }
    GUIProgressEventInfo ::= SEQUENCE {
            subscriberEndpointIRI             EndpointIRI,
            guiEventSubscriptionUuid          BioAPI-UUID OPTIONAL,
            hostingEndpointIRI                EndpointIRI,
            bspProductUuid                    BioAPI-UUID,
            unitID                            BioAPI-UNIT-ID,
            originalBSPHandle                 BioAPI-HANDLE OPTIONAL,
            operation                         BioAPI-GUI-OPERATION,
            suboperation                      BioAPI-GUI-SUBOPERATION,
            purpose                           BioAPI-BIR-PURPOSE,
            moment                            BioAPI-GUI-MOMENT,
            suboperationProgress              UnsignedByte,
            bitmaps                           BioAPI-GUI-BITMAP-ARRAY OPTIONAL,
            text.                             UTF8String OPTIONAL
        }
```

17.4.5 Когда структура получает вызов к функции обратного вызова **BioSPI_GUI_PROGRESS_EVENT_HANDLER** от ПБУ, она должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*incomingNotificationParams*) типа **GUIProgressEvent-NotificationParams** (см. 17.4.3) путем преобразования из параметров вызова функции согласно 17.4.7;

b) проверить таблицу **GUIEventRedirectors** (см. 17.12) на наличие поля, в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** *incomingNotificationParams*, а компонент **guiProgressEventRedirected** имеет значение **TRUE**;

c) в случае, если соответствующее поле отсутствует, создать временное абстрактное значение (*eventInfo*) **GUIProgressEventInfo** (см. 17.4.4), в котором:

1) в случае, если таблица **AttachSessionRemoteReferences** (см. 17.9) имеет поле, в котором компонент **originalBSPHandle** имеет такое же значение, как и компонент **originalBSPHandle** *incomingNotificationParams*, компонент **subscriberEndpointIRI** *eventInfo* устанавливают из компонента **referrerEndpointIRI** такого поля, в противном случае – на ИИР локальной конечной точки;

2) необязательный компонент **guiEventSubscriptionUuid** должен отсутствовать;

3) компонент **hostingEndpointIRI** должен быть установлен на ИИР локальной конечной точки и

4) оставшиеся компоненты должны быть установлены из компонентов *incomingNotificationParams* с такими же именами;

d) в случае, если существует одно соответствующее поле (*redirector*), создать временное абстрактное значение (*eventInfo*) **GUIProgressEventInfo** (см. 17.4.4), в котором:

1) компонент **subscriberEndpointIRI** должен быть установлен из компонента **subscriberEndpointIRI** *redirector*;

2) необязательный компонент **guiEventSubscriptionUuid** должен присутствовать и быть установлен из компонента **guiEventSubscriptionUuid** *redirector*;

3) компонент **hostingEndpointIRI** должен быть установлен на ИИР локальной конечной точки;

4) необязательный компонент **originalBSPHandle** должен отсутствовать и

5) оставшиеся компоненты должны быть установлены из компонентов *incomingNotificationParams* с такими же именами;

e) зарегистрировать операцию прогресса ГИП, основанную на *eventInfo*, на подписчика (либо обработчика операции прогресса ГИП локального приложения, либо главная конечная точка) и определить значение параметра подтверждения (*incomingAcknowledgementParams*) а также определить возвращаемое значение подтверждения (*incomingReturnValue*) согласно разделу 32.

Примечание - Если подписчик на ГИП процесса операций является главной конечной точкой (удаленного приложения), наличие большого изображения может привести к задержке удаленного дисплея. Поэтому рекомендуется ограничить ПБУ при отправке больших изображений.

f) в случае, если *incomingReturnValue* не равно 0, вернуть такое значение ПБУ без выполнения следующих действий;

g) установить исходящие параметры вызова к функции обратного вызова **BioSPI_GUI_PROGRESS_EVENT_HANDLER** путем преобразования из *incomingAcknowledgementParams* согласно 17.4.9; и

h) возвратить ПБУ значение 0.

17.4.6 Когда структура принимает сообщение уведомления ПМО БиоАПИ (см. 13.9) **guiProgressEvent** от второстепенной конечной точки, она должна выполнить следующие действия в указанном порядке:

a) разрешить *incomingNotifcationParams* выступать в качестве значения параметра типа **GUIProgressEvent-NotifcationParams** (см. 17.4.3) сообщения уведомления ПМО БиоАПИ;

b) создать временное абстрактное значение (*eventInfo*) типа **GUIProgressEventInfo** (см. 17.4.4), в котором:

1) компоненты **subscriberEndpointIRI** должны быть установлены на ИИР локальной конечной точки;

2) компонент **hostingEndpointIRI** должен быть установлен из компонента **slaveEndpointIRI** сообщения уведомления ПМО БиоАПИ; и

3) оставшиеся компоненты должны быть установлены из компонентов *incomingNotifcationParams* с такими же именами;

c) зарегистрировать операцию прогресса ГИП, основанную на *eventInfo*, на подписчика (либо обработчика операции прогресса ГИП локального приложения, либо главная конечная точка), и определить значение параметра подтверждения (*incomingAcknowledgementParams*) а также определить возвращаемое значение подтверждения (*incomingReturnValue*) согласно разделу 32;

d) в случае, если *incomingReturnValue* не равно 0, создать и отправить соответствующее сообщение подтверждения ПМО БиоАПИ **guiProgressEvent** (см. 13.5) с возвращаемым значением, установленным на это значение, без выполнения следующих действий;

e) создать и отправить соответствующее сообщение подтверждения ПМО БиоАПИ **guiProgressEvent** (см. 13.5) со значением параметра, установленным на *incomingAcknowledgementParams*, и с возвращаемым значением, установленным на 0.

17.4.7 Преобразование из параметров функции Си обратного вызова **BioSPI_GUI_PROGRESS_EVENT_HANDLER** в тип ACH.1 **GUIProgressEvent-NotifcationParams** (см. 17.4.3) выполняют путем

преобразования из индивидуальных параметров функции в компоненты ACH.1 согласно таблице 132.

Таблица 132 – Преобразование данных из параметров функции обратного вызова **BioSPI_GUI_PROGRESS_EVENT_HANDLER** в тип ACH.1 **GUIProgressEvent-NotifcationParams**

Параметр функции	Компонент типа ACH.1	Раздел, пункт настоящего стандарта
Отсутствует	guiEventSubscriptionUuid	17.4.8
BSPUuid	bspProductUuid	Раздел 19 совместно с 15.58
UnitID	unitID	15.55
BSPHandle	originalBSPHandle	Раздел 19 совместно с 15.42
Operation	operation	15.39
Suboperation	suboperation	15.41
Purpose	purpose	15.14
Moment	moment	15.37
SuboperationProgress	suboperationProgress	15.1.3
Bitmaps	bitmaps	Раздел 19 совместно с 15.35
Text	text	15.2
Response	Отсутствует	Раздел 22

17.4.8 Необязательный компонент **guiEventSubscriptionUuid** должен отсутствовать.

17.4.9 Преобразование из типа ACH.1 **GUIProgressEvent-AcknowledgementParams** (см. 17.4.3) в параметры функции Си обратного

вызова **BioSPI_GUI_PROGRESS_EVENT_HANDLER** выполняют путем преобразования из индивидуальных компонентов АСН.1 в параметры функции согласно таблице 133.

Таблица 133 – Преобразование данных из типа АСН.1 **GUIProgressEvent-AcknowledgementParams** в параметры функции Си обратного вызова **BioSPI_GUI_PROGRESS_EVENT_HANDLER**

Компонент типа АСН.1	Параметр функции	Раздел, подраздел настоящего стандарта
response	Response	Раздел 20 совместно с 15.40

17.4.10 Преобразование из типа АСН.1 **GUIProgressEventHandlerCallbackParams** (см. 17.4.4) в параметры функции Си обратного вызова **BioAPI_GUI_PROGRESS_EVENT_HANDLER** выполняют путем преобразования из индивидуальных компонентов АСН.1 в параметры функции с таблицей 134.

Таблица 134 – Преобразование данных из типа ACH.1 **GUIProgressEventHandlerCallbackParams** в параметры функции Си обратного вызова **BioAPI_GUI_PROGRESS_EVENT_HANDLER**

Компонент типа ACH.1	Параметр функции	Раздел, пункт настоящего стандарта
guiProgressEventHandlerAddress	Отсутствует	17.4.11
guiProgressEventHandlerContext	GUIProgressEventHandlerCtx	15.1.7
bspUuid	BSPUuid	Раздел 19 совместно с 15.58
unitID	UnitID	15.55
bspHandle	BSPHandle	Раздел 19 совместно с 15.42
operation	Operation	15.39
suboperation	Suboperation	15.41
Purpose	Purpose	15.14
moment	Moment	15.37
suboperationProgress	SuboperationProgress	15.1.3
bitmaps	Bitmaps	Раздел 19 совместно с 15.35
text	Text	15.2
Отсутствует	Response	Раздел 22

17.4.11 Компонент **guiProgressEventHandlerAddress** соответствует адресу обратного вызова, а не параметру функции обратного вызова. Преобразование выполняют применяя действия, указанные в 15.1.7.

17.4.12 Преобразование из параметров функции Си обратного вызова **BioAPI_GUI_PROGRESS_EVENT_HANDLER** в тип АСН.1 **GUIProgressEvent-AcknowledgementParams** (см. 17.4.3) выполняют путем преобразования из индивидуальных параметров функции в компоненты АСН.1 согласно таблице 135.

Таблица 135 – Преобразование данных из параметров функции обратного вызова **BioAPI_GUI_PROGRESS_EVENT_HANDLER** в тип АСН.1 **GUIProgressEvent-AcknowledgementParams**

Параметр функции	Компонент типа АСН.1	Раздел, пункт настоящего стандарта
Response	response	Раздел 20 совместно с 15.40

18 Концептуальные таблицы

В настоящем разделе приведено описание определения типа АСН.1 для серии концептуальных таблиц. Указанные типы АСН.1 применяют при спецификации поведения структуры, но их абстрактные значения не появляются при каком-либо обмене сообщениями ПМО БиоАПИ между конечными точками ПМО БиоАПИ, в поэтому их не кодируют. Для поддержки таблиц, описанных в настоящем разделе, необходима соответствующая главная точка ПМО БиоАПИ, а для второстепенных точек необходима соответствующая второстепенная точка ПМО БиоАПИ. Конечная точка ПМО БиоАПИ может использовать любое подходящее представление для концептуальных таблиц и не должна поддерживать преобразование содержания таблиц в последовательную форму.

Примечание – Реализация конечной точки ПМО БиоАПИ может использовать поддержку такого преобразования в последовательную форму для целей администрирования и отладки, но это не является обязательным условием. Примером такой способности

является управление интерфейсом, который поддерживает текущее содержание концептуальных таблиц.

18.1 Концептуальная таблица MasterEndpoints

Данная концептуальная таблица представлена во всех второстепенных конечных точках и определена в АСН.1 следующим образом:

```
MasterEndpoints ::= SET OF endpoint MasterEndpoint
        MasterEndpoint ::= SEQUENCE {
        masterEndpointIRI     EndpointIRI
    }
```

18.1.1 Общие положения

Поле данной таблицы представляет собой главную точку, которая является конечной точки ПМО БиоАПИ и связана с локальной конечной точкой.

18.1.2 Компоненты

18.1.2.1 Компонент **masterEndpointIRI** должен содержать ИИР конечной точки главной конечной точки.

18.1.2.2 Не допускается два или более полей с одинаковыми значениями компонента **masterEndpointIRI**.

18.1.3 Удаление поля

18.1.3.1 Данный пункт применяют только в том случае, если на него имеется ссылка в других пунктах настоящего стандарта, когда поле таблицы **MasterEndpoints** должно быть удалено.

18.1.3.2 Разрешить *masterEndpointIRI* выступать в качестве значения компонента **masterEndpointIRI** того поля, которое должно быть удалено.

18.1.3.3 Для каждого поля таблицы **RunningBSPRemoteReferences** (см. 18.6), в котором компонент **referrerEndpointIRI** имеет значение *masterEndpointIRI*, структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*bspUnloadCallParams*) типа **BSPUnloadCallParams** (см. 16.10.3), в котором:

1) компонент **bspUuid** должны быть установлены из компонента **bspProductUuid** поля и

2) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** должны быть установлены на 0;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPUnload** (см. 16.10), в котором параметры вызова функции установлены путем преобразования из *bspUnloadCallParams* согласно 16.10.3;

c) удалить поле таблицы **RunningBSPRemoteReferences**, (выполняя действия, указанные в 18.6.3).

18.1.3.4 Для каждого поля таблицы **UnitEventNotifcationDisablers** (см. 18.7), в котором компонент **referrerEndpointIRI** имеет значение **masterEndpointIRI**, структура удаляет поле таблицы **UnitEventNotifcationDisablers**, (выполняя действия, указанные в 18.7.3).

18.1.3.5 Для каждого поля таблицы **GUIEventRemoteSubscriptions** (см. 18.11), в котором компонент **subscriberEndpointIRI** имеет значение *masterEndpointIRI*, структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*unsubscribeFromGUIEventsCallParams*) типа **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** установлен из необязательного компонента **guiEventSubscriptionUuid** поля (присутствие и значение);

2) в случае, если необязательный компонент **originalBSPHandle** поля отсутствует, необязательный компонент **bspUuid** из *unsubscribeFromGUIEventsCallParams* должен быть установлен из

компонента **bspProductUuid** поля; в противном случае он должен отсутствовать;

3) необязательный компонент **bspHandle** установлен из необязательного компонента **originalBSPHandle** поля (присутствие и значение);

4) в случае, если компонент **guiSelectEventSubscribed** поля имеет значение **TRUE**, компонент **guiSelectEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать используемому и должен быть установлен в определенный реализацией адрес памяти отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0;

5) в случае, если компонент **guiStateEventSubscribed** поля имеет значение **TRUE**, компонент **guiStateEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0;

6) в случае, если компонент **guiProgressEventSubscribed** поля имеет значение **TRUE**, компонент **guiProgressEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0 и

7) компоненты **guiSelectEventHandlerContext**, **guiStateEventHandlerContext** и **guiProgressEventHandlerContext** устанавливают на 0;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnsubscribeFromGUIEvents** (см. 16.23), в котором параметры вызова функции устанавливают путем преобразования из *unsubscribeFromGUIEventsCallParams* согласно 16.23.5;

c) удалить поле таблицы **GUIEventRemoteSubscriptions** (выполняя действия, указанные в 18.11.3).

18.1.3.6 Для каждого поля таблицы **GUIEventRedirectors** (см. 18.12), в котором компонент **subscriberEndpointIRI** имеет значение *masterEndpointIRI*, структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*unredirectCallParams*) типа **UnredirectGUIEvents-RequestParams** (см. 16.29.2), в котором все компоненты устанавливают из компонентов поля с такими же названиями;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnredirectGUIEvents** (см. 16.29), в котором параметры вызова функции устанавливают путем преобразования из *unredirectCallParams* согласно 16.29.5;

c) удалить поле таблицы **GUIEventRedirectors**, выполняя действия, указанные в 18.12.3.

18.1.4 Жизненный цикл

18.1.4.1 Поле может быть добавлено в таблицу **MasterEndpoints** в случае, когда структура получает сообщение запроса ПМО БиоАПИ **addMaster** от конечной точки ПМО БиоАПИ (см. 16.4.2).

18.1.4.2 Поле таблицы **MasterEndpoints** может быть удалено в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_Terminate** от локального приложения (см. 16.3) и

b) когда структура получает сообщение запроса ПМО БиоАПИ **deleteMaster** от конечной точки ПМО БиоАПИ (см. 16.5.2).

18.2 Концептуальная таблица VisibleEndpoints

Данная концептуальная таблица представлена во всех конечных точках ПМО БиоАПИ и определена в АСН.1 следующим образом:

```
VisibleEndpoints ::= SET OF endpoint VisibleEndpoint
VisibleEndpoint ::= BioAPI-FRAMEWORK-SCHEMA
```

18.2.1 Общие положения

18.2.1.1 Поле данной таблицы представляет собой видимую конечную точку, которая является конечной точкой ПМО БиоАПИ, чьи компоненты (структура, ПБУ и ПБФ) являются видимыми для локального приложения.

18.2.1.2 Данная таблица имеет одно поле для локальной точки и по одному полю для каждой второстепенной точки локальной конечной точки (ноль или более). Каждое поле содержит копию схемы структуры, которая представлена в реестре компонентов (локальной или второстепенной) конечной точки ПМО БиоАПИ.

18.2.2 Компоненты

18.2.2.1 Компонент **hostingEndpointIRI** должен содержать ИИР конечной точки видимой конечной точки. Видимая конечная точка должна быть либо локальной конечной точкой либо второстепенной конечной точкой.

18.2.2.2 Другие компоненты должны содержать копию атрибутов схемы структуры в реестре компонентов конечной точки ПМО БиоАПИ.

18.2.2.3 Не должно быть два или более полей с одинаковыми значениями компонента **hostingEndpointIRI**.

18.2.3 Удаление поля

18.2.3.1 Данный пункт применяют только в том случае, если на него имеются ссылки в других пунктах настоящего стандарта, когда поле таблицы **VisibleEndpoints** должно быть удалено.

18.2.3.2 Разрешить *hostingEndpointIRI* выступать в качестве значения компонента **hostingEndpointIRI** того поля, которое должно быть удалено.

18.2.3.3 Каждое поле таблицы **VisibleBSPRegistrations** (см. 18.3), в котором компонент **hostingEndpointIRI** имеет значение *hostingEndpointIRI*, структура должна удалить, выполняя действия, указанные в 18.3.3.

18.2.3.4 Каждое поле таблицы **VisibleBFPRegistrations** (см. 18.4), в котором компонент **hostingEndpointIRI** имеет значение *hostingEndpointIRI*, структура должна удалить, выполняя действия, указанные в 18.4.3.

18.2.4 Жизненный цикл

18.2.4.1 Поле может быть добавлено в таблицу **VisibleEndpoints** в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_Init** или **BioAPI_InitEndpoint** от локального приложения (см. 16.1) и

b) когда структура получает вызов к функции **BioAPI_LinkToEndpoint** от локального приложения (см. 16.4).

18.2.4.2 Поле таблицы **VisibleEndpoints** может быть удалено в следующих случаях:

a) когда структура получает сообщение уведомления ПМО БиоАПИ **masterDeletionEvent** от второстепенной конечной точки (см. 16.3.2) и

b) когда структура получает вызов к функции **BioAPI_UnlinkFromEndpoint** от локального приложения (см. 16.5).

18.3 Концептуальная таблица VisibleBSPRegistrations

Данная концептуальная таблица представлена во всех конечных точках ПМО БиоАПИ и определен в АСН.1 следующим образом:

```
VisibleBSPRegistrations ::= SET OF
        registration VisibleBSPRegistration

VisibleBSPRegistration ::= BioAPI-BSP-SCHEMA
```

18.3.1 Общие положения

18.3.1.1 Поле данной таблицы представляет собой ПБУ, зарегистрированный в реестре компонентов видимой конечной точки.

18.3.1.2 Данная таблица имеет по одному полю для каждого ПБУ, зарегистрированного в локальном реестре компонентов, и по одному полю для каждого ПБУ, зарегистрированного в локальном реестре компонентов каждой второстепенной конечной точки локальной конечной точки. Каждое поле содержит копию схемы ПБУ, которая представлена в реестре компонентов (локальной или второстепенной) конечной точки ПМО БиоАПИ

18.3.1.3 Если ПБУ зарегистрирован в реестрах компонентов двух или более конечных точек, в таблице должны быть выделены поля для каждой пары ПБУ/конечная точка, которые должны различаться, по крайней мере, значениями компонента **hostingEndpointIRI**.

18.3.1.4 Данная таблица также поддерживает трансляцию между УУИД продукта ПБУ и УУИД доступа ПБУ.

Примечание – УУИД продукта ПБУ, который идентифицирует ПБУ в качестве программного продукта, устанавливает разработчик ПБУ и должен быть включен в зарегистрированную схему ПБУ. Как правило, УУИД продукта ПБУ не различается при проведении множества регистраций одного и того же ПБУ в различных реестрах компонентов. УУИД доступа ПБУ динамически создается структурой и устанавливается для каждого ПБУ, представленного в таблице, и он не виден для локального приложения. Если в структуре присутствуют две или более второстепенные конечные точки, один и тот же ПБУ может быть зарегистрирован в двух или более видимых конечных точках, и в таблице может быть два или более полей, содержащих один и тот же УУИД продукта ПБУ. В данном случае, использование УУИД продукта ПБУ в вызове к **BioAPI_BSPLoad** не будет достаточным для однозначной идентификации и загружаемого ПБУ и конечной точки ПМО БиоАПИ, в которую ПБУ должен быть загружен. Но УУИД доступа ПБУ может быть использован в любом вызове к **BioAPI_BSPLoad**. С другой стороны, УУИД продукта ПБУ может использоваться в вызове к **BioAPI_BSPLoad** до тех пор, пока ПБУ только единожды представлен в таблице, так как УУИД продукта ПБУ является достаточным для идентификации и загружаемого ПБУ и конечной точки ПМО БиоАПИ, в которую ПБУ должен быть загружен.

18.3.2 Компоненты

18.3.2.1 Компонент **hostingEndpointIRI** должен содержать ИИР конечной точки ПМО БиоАПИ, реестр компонентов которой содержит схему ПБУ зарегистрированного ПБУ. Конечная точка ПМО БиоАПИ должна быть либо локальной конечной точкой либо второстепенной конечной точкой

18.3.2.2 Компонент **bspAccessUuid** должен содержать УУИД доступа ПБУ, который динамически создан структурой для идентификации зарегистрированного ПБУ.

18.3.2.3 Другие компоненты должны содержать копию атрибутов схемы ПБУ в реестре компонентов конечной точки ПМО БиоАПИ.

18.3.2.4 Не должно быть два или более полей с одинаковыми значениями двух компонентов **hostingEndpointIRI** и **bspProductUuid**.

18.3.2.5 Не должно быть два или более полей с одинаковыми значениями компонента **bspAccessUuid**.

18.3.2.6 Значение компонента **bspAccessUuid** в поле должно отличаться от значения компонента **bspProductUuid** в этом или в каком-либо другом поле.

18.3.3 Удаление поля

18.3.3.1 Данный пункт применяют только в том случае, если на него есть ссылка в других пунктах настоящего стандарта, когда поле таблицы **VisibleBSPRegistrations** должно быть удалено.

18.3.3.2 Разрешить *hostingEndpointIRI* выступать в качестве значения компонента **hostingEndpointIRI** и *bspProductUuid* в качестве значения компонента **bspProductUuid** того поля, которое должно быть удалено.

18.3.3.3 Для каждого поля таблицы **RunningBSPLocalReferences** (см. 18.5), в котором компонент **hostingEndpointIRI** имеет значение *hostingEndpointIRI* и компонент **bspProductUuid** имеет значение *bspProductUuid*, структура должна выполнить следующие действия в указанном порядке:

a) в случае, если *hostingEndpointIRI* не является ИИР локальной конечной точки, удалить поле таблицы **RunningBSPLocalReferences**, выполняя действия, указанные в 18.5.3, без выполнения следующих действий;

b) создать временное абстрактное значение (*bspUnloadCallParams*) типа **BSPUnloadCallParams** (см. 16.10.3), в котором:

1) компонент **bspUuid** устанавливают на *bspProductUuid* и

2) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** должны быть установлены на 0;

c) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPUnload** (см. 16.10), в котором параметры вызова функции устанавливают путем преобразования из *bspUnloadCallParams* согласно 16.10.3;

d) удалить поле таблицы **RunningBSPLocalReferences**, выполняя действия, указанные в 18.5.3.

18.3.3.4 Если *hostingEndpointIRI* не является ИИР локальной конечной точки, тогда, для каждого поля таблицы **RunningBSPRemoteReferences** (см. 18.6), в котором компонент **bspProductUuid** имеет значение *bspProductUuid*, структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*bspUnloadCallParams*) типа **BSPUnloadCallParams** (см. 18.10.3), в котором:

1) компонент **bspUuid** устанавливают на *bspProductUuid*; и

2) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** устанавливают на 0;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPUnload** (см. 16.10), в котором параметры вызова функции устанавливают путем преобразования из *bspUnloadCallParams* согласно 16.10.3;

c) удалить поле таблицы **RunningBSPRemoteReferences**, выполняя действия, указанные в 18.6.3.

18.3.3.5 Для каждого поля таблицы **GUIEventLocalSubscriptions** (см. 18.10), в котором компонент **hostingEndpointIRI** имеет значение *hostingEndpointIRI* и компонент **bspProductUuid** имеет значение *bspProductUuid*, структура должна выполнить следующие действия в указанном порядке:

a) в случае, если *hostingEndpointIRI* не является ИИР локальной конечной точки, удалить поле таблицы **GUIEventLocalSubscriptions**, выполняя действия, указанные в 18.10.3, без выполнения следующих действий;

b) создать временное абстрактное значение (*unsubscribeFromGUIEventsCallParams*) типа **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** устанавливают из необязательного компонента **guiEventSubscriptionUuid** поля (присутствие и значение);

2) в случае, если необязательный компонент **originalBSPHandle** поля отсутствует, необязательный компонент **bspUuid** из *unsubscribeFromGUIEventsCallParams* устанавливают из компонента **bspProductUuid** поля; в противном случае, он должен отсутствовать;

3) необязательный компонент **bspHandle** устанавливают из необязательного компонента **originalBSPHandle** поля (присутствие и значение);

4) Если компонент **guiSelectEventHandlerAddress** поля имеет значение, отличающееся от 0, компонент **guiSelectEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который

должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0;

5) в случае, если компонент **guiStateEventHandlerAddress** поля имеет значение, отличающееся от 0, компонент **guiStateEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0;

6) Если компонент **guiProgressEventHandlerAddress** поля имеет значение, отличающееся от 0, компонент **guiProgressEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0; and

7) компоненты **guiSelectEventHandlerContext**, **guiStateEventHandlerContext**, и **guiProgressEventHandlerContext** устанавливают на 0;

c) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnsubscribeFromGUIEvents** (см. 16.23), в котором параметры вызова функции устанавливают путем преобразования из *unsubscribeFromGUIEventsCallParams* согласно 16.23.6;

d) удалить поле таблицы **GUIEventLocalSubscriptions**, выполняя действия, указанные в 18.10.3;

18.3.3.6 Если *hostingEndpointIRI* не является ИИР локальной конечной точки, тогда, для каждого поля таблицы **GUIEventRemoteSubscriptions** (см. 18.11), в котором компонент **bspProductUuid** имеет значение *bspProductUuid*, структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*unsubscribeFromGUIEventsCallParams*) типа **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** устанавливают из необязательного компонента **guiEventSubscriptionUuid** поля (присутствие и значение);

2) в случае, если необязательный компонент **originalBSPHandle** поля отсутствует, необязательный компонент **bspUuid** из *unsubscribeFromGUIEventsCallParams* устанавливают из компонента **bspProductUuid** поля; в противном случае компонент должен отсутствовать;

3) необязательный компонент **bspHandle** устанавливают из необязательного компонента **originalBSPHandle** поля (присутствие и значение);

4) в случае, если компонент **guiSelectEventSubscribed** поля имеет значение **TRUE**, компонент **guiSelectEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0;

5) в случае, если компонент **guiStateEventSubscribed** поля имеет значение **TRUE**, компонент **guiStateEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0;

6) в случае, если компонент **guiProgressEventSubscribed** поля имеет значение **TRUE**, компонент **guiProgressEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в

определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливают на 0 и

7) компоненты **guiSelectEventHandlerContext**, **guiStateEventHandlerContext** и **guiProgressEventHandlerContext** устанавливают на 0;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnsubscribeFromGUIEvents** (см. 16.23), в котором параметры вызова функции устанавливают путем преобразования из *unsubscribeFromGUIEventsCallParams* согласно 16.23.6;

c) удалить поле таблицы **GUIEventRemoteSubscriptions**, выполняя действия, указанные в 18.11.3.

18.3.4 Жизненный цикл

18.3.4.1 Поле может быть добавлено в таблицу **VisibleBSPRegistrations** в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_Init** or **BioAPI_InitEndpoint** от локального приложения (см. 16.1);

b) когда структура получает вызов к функции **BioAPI_LinkToEndpoint** от локального приложения (см. 16.4);

c) когда структура получает вызов к функции **BioAPI_RegisterBSP** от локального приложения (см. 16.59);

d) когда структура получает сообщение запроса ПМО БиоАПИ **registerBSP** от главной конечной точки (см. 16.59.2) и

e) когда структура получает сообщение уведомления ПМО БиоАПИ **bspRegistrationEvent** от второстепенной конечной точки (см. 16.59.2).

18.3.4.2 Поле таблицы **VisibleBSPRegistrations** может быть удалено в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_RegisterBSP** от локального приложения (см. 16.59);

b) когда структура получает сообщение запроса ПМО БиоАПИ **registerBSP** от главной конечной точки (см. 16.59.2);

c) когда структура получает сообщение уведомления ПМО БиоАПИ **bspRegistrationEvent** от второстепенной конечной точки (см. 16.59.2);

d) когда структура получает вызов к функции **BioAPI_UnregisterBSP** от локального приложения (см. 16.60);

e) когда структура получает сообщение запроса ПМО БиоАПИ **unregisterBSP** от главной конечной точки (см. 16.60.2);

f) когда структура получает сообщение уведомления ПМО БиоАПИ **bspUnregistrationEvent** от второстепенной конечной точки (см. 16.60.2) и

g) когда поле таблицы **VisibleEndpoints** удалено (см. 18.2.4.2).

18.4 Концептуальная таблица VisibleBFPRegistrations

Данная концептуальная таблица представлена во всех конечных точках ПМО БиоАПИ и определена в АСН.1 следующим образом:

```
VisibleBFPRegistrations ::= SET OF
        registration VisibleBFPRegistration

VisibleBFPRegistration ::= BioAPI-BFP-SCHEMA
```

18.4.1 Общие положения

18.4.1.1 Поле данной таблицы представляет ПБФ как зарегистрированный в реестре компонентов видимой конечной точки и, таким образом, как доступный для использования локальным приложением через зарегистрированный в той же локальной точке ПБУ.

18.4.1.2 Данная таблица имеет по одному полю для каждого ПБФ, зарегистрированного в локальном реестре компонентов, и по одному полю для каждого ПБФ, зарегистрированного в локальном реестре компонентов каждой второстепенной конечной точки локальной конечной точки. Каждое поле содержит копию схемы ПБФ, которая представлена в реестре компонентов (локальной или второстепенной) конечной точки ПМО БиоАПИ.

18.4.1.3 Если ПБФ зарегистрирован в реестрах компонентов двух или более конечных точек, таблица выделяет поле для каждой пары ПБФ/конечная точка: такие поля будут различаться, по крайней мере, значениями компонента **hostingEndpointIRI**.

18.4.2 Компоненты

18.4.2.1 Компонент **hostingEndpointIRI** должен содержать ИИР конечной точки ПМО БиоАПИ, реестр компонентов которой содержит схему ПБФ зарегистрированного ПБФ. Конечная точка ПМО БиоАПИ должна быть либо локальной конечной точкой либо второстепенной конечной точкой.

18.4.2.2 Другие компоненты должны содержать копию атрибутов схемы ПБФ в реестре компонентов конечной точки ПМО БиоАПИ.

18.4.2.3 Не должно быть два или более полей с одинаковыми значениями двух компонентов **hostingEndpointIRI** и **bfpProductUuid**.

18.4.3 Удаление поля

18.4.3.1 Данный пункт применяют только в том случае, если на него имеется ссылка в других пунктах настоящего стандарта, когда поле таблицы **VisibleBFPRegistrations** должно быть удалено.

18.4.3.2 Дополнительных действий не требуется.

18.4.4 Жизненный цикл

18.4.4.1 Поле может быть добавлено в таблицу **VisibleBFPRegistrations** в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_Init** or **BioAPI_InitEndpoint** от локального приложения (см. 16.1);

b) когда структура получает вызов к функции **BioAPI_LinkToEndpoint** от локального приложения (см. 16.4);

c) когда структура получает вызов к функции **BioAPI_RegisterBFP** от локального приложения (см. 16.61);

d) когда структура получает сообщение запроса ПМО БиоАПИ **registerBFP** от главной конечной точки (см. 16.61.2) и

e) когда структура получает сообщение уведомления ПМО БиоАПИ **bfpRegistrationEvent** от второстепенной конечной точки (см. 16.61.2).

18.4.4.2 Поле **VisibleBFPRegistrations** может быть удалено в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_RegisterBFP** от локального приложения (см. 16.61);

b) когда структура получает сообщение запроса ПМО БиоАПИ **registerBFP** от главной конечной точки (см. 16.61.2);

c) когда структура получает сообщение уведомления ПМО БиоАПИ **bfpRegistrationEvent** от второстепенной конечной точки (см. 16.61.2);

d) когда структура получает вызов к функции **BioAPI_UnregisterBFP** от локального приложения (см. 16.62);

e) когда структура получает сообщение запроса ПМО БиоАПИ **unregisterBFP** от главной конечной точки (см. 16.62.2);

f) когда структура получает сообщение уведомления ПМО БиоАПИ **bfpUnregistrationEvent** от второстепенной конечной точки (см. 16.62.2) и

g) когда поле таблицы **VisibleEndpoints** удалено (см. 18.2.4.2).

18.5 Концептуальная таблица RunningBSPLocalReferences

Данная концептуальная таблица представлена во всех конечных точках ПМО БиоАПИ и определена в АСН.1 следующим образом:

```
RunningBSPLocalReferences ::= SET OF
                reference RunningBSPLocalReference

RunningBSPLocalReference ::= SEQUENCE {
                hostingEndpointIRI EndpointIRI,
                bspProductUuid BioAPI-UUID,
                useBSPAccessUuid BOOLEAN,
                unitEventHandlerAddress MemoryAddress,
```

unitEventHandlerContext MemoryAddress

}

18.5.1 Общие положения

18.5.1.1 Поле данной таблицы представляет собой активную ПБУ локальную связь, состоящую из:

a) связи, установленной вызовом функции **BioAPI_BSPLoad**, и удерживаемой локальным приложением с ПБУ, активным в локальной конечной точке или во второстепенной конечной точке и

b) обязательства структуры по совершению обратного вызова к обработчику модуля операций локального приложения на определенные поступающие уведомления модуля операций (либо обратный вызов уведомления модуля операций от ПБУ, либо сообщения уведомления модуля операций, связанных с ПБУ), (необязательно).

18.5.1.2 Входящий вызов функции **BioAPI_BSPLoad** может создать новое поле активной локальной ПБУ связи даже в том случае, если параметры вызова аналогичными предыдущему входящему вызову функции **BioAPI_BSPLoad**. Входящий вызов функции **BioAPI_BSPUnload**, параметры которого совпадают с существующим полем активной локальной ПБУ связи, может удалить такое поле. В случае множественных совпадений (множество идентичных полей), одно любое поле из совпадающих может быть удалено.

18.5.1.3 Активная ПБУ локальная связь может использовать либо УУИД продукта ПБУ заданного ПБУ, либо динамически определенного структурой для такого ПБУ УУИД доступа ПБУ. Локальное приложение может приобрести УУИД доступа ПБУ заданного ПБУ путем вызова функции **BioAPI_EnumBSPs**.

18.5.1.4 Как правило, активный ПБУ не выгружается до тех пор, пока локальное приложение (если есть) удерживает активную ПБУ локальную связь с ним, либо пока главная конечная точка удерживает активную ПБУ удаленную связь с ним.

18.5.2 Компоненты

18.5.2.1 Компонент **hostingEndpointIRI** должен содержать ИИР конечной точки ПМО БиоАПИ, включающей в себя зарегистрированный ПБУ, который относится к активной ПБУ локальной связи. Конечная точка ПМО БиоАПИ должна быть либо локальной, либо второстепенной конечной точкой

18.5.2.2 Компонент **bspProductUuid** должен содержать УУИД продукта ПБУ зарегистрированного ПБУ.

18.5.2.3 Компонент **useBSPAccessUuid** должен определять, предоставило ли локальное приложение УУИД доступа ПБУ (а не УУИД продукта ПБУ) для идентификации зарегистрированного ПБУ в вызове функции **BioAPI_BSPLoad** (см. 16.9), который вызвал добавление поля.

Примечание – При последующем вызове функции **BioAPI_BSPUnload** (см. 16.10) должен быть предоставлен аналогичный УУИД (или УУИД доступа ПБУ или УУИД продукта ПБУ) для совпадения с указанной активной ПБУ локальной связью. При обратном вызове уведомления модуля операций, который создается с использованием указанной активной ПБУ локальной связи, должен быть предоставлен УУИД, аналогичный входящему в обработчик модуля операций локального приложения.

18.5.2.4 Компонент **unitEventHandlerAddress** должен содержать адрес обратного вызова обработчика модуля событий локального приложения. Значение, отличающееся от 0, означает обязательство структуры по совершению обратного вызова к обработчику модуля операций локального приложения с использованием такого адреса обратного вызова.

18.5.2.5 Компонент **unitEventHandlerContext** должен содержать адрес контекста, который должен быть передан как входные данные в обработчик модуля операций локального приложения.

18.5.2.6 Возможно наличие нескольких полей с одинаковыми значениями одного или более (или всех) их компонентов.

18.5.3 Удаление поля

18.5.3.1 Данный пункт применяют только в том случае, если на него имеется ссылка в других пунктах настоящего стандарта в случаях, когда поле таблицы **RunningBSPLocalReferences** необходимо удалить.

18.5.3.2 Разрешить *hostingEndpointIRI* выступать в качестве значения компонента **hostingEndpointIRI** и *bspProductUuid* выступать в качестве значения компонента **bspProductUuid** того поля, которое должно быть удалено.

18.5.3.3 Если обнаруженное поле является единственным полем таблицы **RunningBSPLocalReferences**, в котором компонент **hostingEndpointIRI** имеет значение *hostingEndpointIRI* и компонент **bspProductUuid** имеет значение *bspProductUuid*, для каждого поля таблицы **AttachSessionLocalReferences** (см. 18.8), в котором компонент **hostingEndpointIRI** имеет значение *hostingEndpointIRI* и компонент **bspProductUuid** имеет значение *bspProductUuid*, структура должна выполнить следующие действия в указанном порядке:

a) в случае, если *hostingEndpointIRI* не является ИИР локальной конечной точки, удалить поле таблицы **AttachSessionLocalReferences**, выполняя действия, указанные в 18.8.3, без выполнения следующих действий;

b) создать временное абстрактное значение (читай *bspDetachCallParams*) типа **BSPDetach-RequestParams** (см. 16.14.2), в котором компонент **originalBSPHandle** устанавливается из компонента **originalBSPHandle** поля;

c) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPDetach** (см. 16.14), в котором параметры вызова функции должны быть установлены путем преобразования из *bspDetachCallParams* согласно 16.14.5;

d) удалить поле таблицы **AttachSessionLocalReferences**, выполняя действия, указанные в 18.8.3.

18.5.4 Жизненный цикл

18.5.4.1 Поле может быть добавлено в таблицу **RunningBSPLocalReferences** в следующем случае:

– когда структура получает вызов к функции **BioAPI_BSPLoad** от локального приложения (см. 16.9).

18.5.4.2 Поле таблицы **RunningBSPLocalReferences** может быть удалено в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_BSPUnload** от локального приложения (см. 16.10); и

b) когда поле таблицы **VisibleBSPRegistrations** удалено (см. 18.3.4.2).

18.6 Концептуальная таблица RunningBSPRemoteReferences

Данная концептуальная таблица представлена во всех второстепенных конечных точках и определена в АСН.1 следующим образом:

```
RunningBSPRemoteReferences ::= SET OF
                reference RunningBSPRemoteReference

RunningBSPRemoteReference ::= SEQUENCE {
                referrerEndpointIRI EndpointIRI,
                bspProductUuid BioAPI-UUID,
                unitEventSubscription BOOLEAN
        }
```

18.6.1 Общие положения

18.6.1.1 Поле данной таблицы представляет собой активную ПБУ удаленную связь, состоящую из:

a) связи, установленной сообщением запроса ПМО БиоАПИ **bspLoad** и удерживаемой главной конечной точкой с ПБУ, активным в локальной конечной точке и

b) обязательства структуры по отправлению сообщения уведомления **unitEvent** к главной конечной точке на определенные входящие обратные вызовы уведомлений модуля операций от ПБУ (необязательно).

18.6.1.2 Поступающее сообщение запроса ПМО БиоАПИ **bspLoad** может создать новое поле активной удаленной ПБУ связи даже в том случае, если параметры значения сообщение запроса ПМО БиоАПИ являются аналогичными предыдущему поступившему сообщению запроса ПМО БиоАПИ **bspLoad**. Поступающее сообщение запроса ПМО БиоАПИ **bspLoad**, значение параметра которого (дополнительно к ИИР конечной точки отправляющей сообщение конечной точки) совпадает с существующим полем активной удаленной ПБУ связи, может удалить такое поле. В случае множественных совпадений (множество идентичных полей), одно поле из совпадающих может быть удалено.

18.6.1.3 Как правило, активный ПБУ не выгружается до тех пор, пока локальное приложение (если есть) удерживает активную ПБУ локальную связь с ним, либо пока главная конечная точка удерживает активную ПБУ удаленную связь с ним.

18.6.2 Компоненты

18.6.2.1 Компонент **referrerEndpointIRI** должен содержать ИИР конечной точки главной конечной точки, которая удерживает активную ПБУ удаленную связь. Она не должна быть локальной конечной точкой.

18.6.2.2 Компонент **bspProductUuid** должен содержать УУИД продукта ПБУ зарегистрированного ПБУ.

18.6.2.3 Компонент **unitEventSubscription** должен определять наличие обязательства структуры по правлению сообщения уведомления **unitEvent** к главной конечной точке.

18.6.2.4 Допускается наличие нескольких полей с одинаковыми значениями одного или более (или всех) их компонентов.

18.6.3 Удаление поля

18.6.3.1 Данный пункт применяют только в том случае, если на него имеется ссылка в других пунктах настоящего стандарта, когда поле таблицы **RunningBSPRemoteReferences** должно быть удалено.

18.6.3.2 Разрешить *referrerEndpointIRI* выступать в качестве значения компонента **referrerEndpointIRI**, а *bspProductUuid* – в качестве значения компонента **bspProductUuid** того поля, которое должно быть удалено.

18.6.3.3 Если обнаруженное поле является единственным полем таблицы **RunningBSPRemoteReferences**, в котором компонент **referrerEndpointIRI** имеет значение *referrerEndpointIRI* и компонент **bspProductUuid** имеет значение *bspProductUuid*, для каждого поля таблицы **AttachSessionRemoteReferences** (см. 18.9), в котором компонент **referrerEndpointIRI** имеет значение *referrerEndpointIRI* и компонент **bspProductUuid** имеет значение *bspProductUuid*, структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*bspDetachCallParams*) типа **BSPDetach-RequestParams** (см. 16.14.2), в котором компонент **originalBSPHandle** устанавливается из компонента **originalBSPHandle** поля;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_BSPDetach** (см. 16.14), в котором параметры вызова функции должны быть установлены путем преобразования из *bspDetachCallParams* согласно 16.14.5;

c) удалить поле таблицы **AttachSessionRemoteReferences**, выполняя действия, указанные в 18.9.3.

18.6.4 Жизненный цикл

18.6.4.1 Поле может быть добавлено в таблицу **RunningBSPRemoteReferences** в следующем случае:

– когда структура получает сообщение запроса ПМО БиоАПИ **bspLoad** от главной конечной точки (см. 16.9.2).

18.6.4.2 Поле таблицы **RunningBSPRemoteReferences** может быть удалено в следующих случаях:

a) когда структура получает сообщение запроса ПМО БиоАПИ **bspUnload** от главной конечной точки (см. 16.10.2);

b) когда поле таблицы **MasterEndpoints** удалено (см. 18.1.4.2) и

c) когда поле таблицы **VisibleBSPRegistrations** удалено (см. 18.3.4.2).

18.7 Концептуальная таблица UnitEventNotificationDisablers

Данная концептуальная таблица представлена во всех конечных точках и определена в АСН.1 следующим образом:

```
UnitEventNotificationDisablers ::= SET OF
                disabler UnitEventNotificationDisabler

UnitEventNotificationDisabler ::= SEQUENCE {
                referrerEndpointIRI EndpointIRI,
                bspProductUuid BioAPI-UUID,
                unitEventTypes BioAPI-UNIT-EVENT-TYPE-MASK
        }
```

18.7.1 Общие положения

18.7.1.1 Поле данной таблицы представляет собой отключатель операции уведомления, который состоит из запрета для структуры либо на отправление сообщения уведомления **unitEvent** к определенной главной точке, либо на вызов любого обработчика модуля операций локального приложения (в зависимости от значения **referrerEndpointIRI**) для определенных ПБУ или для определенных типов операций.

18.7.1.2 Поле добавляется в данную таблицу в результате входящего вызова к **BioAPI_EnableEventNotifications** или поступающего сообщения запроса ПМО БиоАПИ **enableEventNotifications**. Тем не менее, один и тот же входящий вызов или сообщение запроса ПМО БиоАПИ могут также стать причиной усовершенствования или удаления поля таблицы.

18.7.1.3 Хотя каждое поле данной таблицы содержит УУИД продукта ПБУ, добавление, усовершенствование и удаление полей таблицы полностью не зависит от загрузки или выгрузки ПБУ.

18.7.2 Компоненты

18.7.2.1 Компонент **referrerEndpointIRI** должен содержать либо ИИР конечной точки локальной конечной точки, либо ИИР конечной точки главной конечной точки.

18.7.2.2 Компонент **bspProductUuid** должен содержать УУИД продукта ПБУ зарегистрированного ПБУ, для которого один или более типов модулей операций уведомления отключены.

18.7.2.3 Компонент **unitEventTypes** должен определять один или более типов модулей операций уведомления, которые должны быть отключены. Этот компонент не должен иметь значение, указывающее на то, что все модули операций уведомления включены.

18.7.2.4 Не должно быть два или более полей с одинаковыми значениями компонентов **referrerEndpointIRI** и **bspProductUuid**.

18.7.3 Удаление поля

18.7.3.1 Данный пункт применяют только в том случае, если на него имеется ссылка в других пунктах настоящего стандарта, когда поле таблицы **UnitEventNotificationDisablers** должно быть удалено.

18.7.3.2 Дополнительных действий не требуется.

18.7.4 Жизненный цикл

18.7.4.1 Поле может быть добавлено в таблицу **UnitEventNotifcationDisablers** в следующих случаях:

a) когда структура получает сообщение запроса ПМО БиоАПИ **enableEventNotifcations** от главной конечной точки (см. 16.16.5) и

b) когда структура получает вызов к функции **BioAPI_EnableEventNotifcations** от локального приложения (см. 16.16.4).

18.7.4.2 Поле таблицы **UnitEventNotifcationDisablers** может быть удалено в следующих случаях:

a) когда структура получает сообщение запроса ПМО БиоАПИ **enableEventNotifcations** от главной конечной точки (см. 16.16.5);

b) когда структура получает вызов к функции **BioAPI_EnableEventNotifcations** от локального приложения (см. 16.16.4) и

c) когда поле таблицы **MasterEndpoints** удаляется (см. 19.1.3).

18.8 Концептуальная таблица AttachSessionLocalReferences

Данная концептуальная таблица представлена во всех конечных точках ПМО БиоАПИ и определена в АСН.1 следующим образом:

```
AttachSessionLocalReferences ::= SET OF
                reference AttachSessionLocalReference

AttachSessionLocalReference ::= SEQUENCE {
                hostingEndpointIRI EndpointIRI,
                bspProductUuid BioAPI-UUID,
                useBSPAccessUuid BOOLEAN,
                originalBSPHandle BioAPI-HANDLE,
                localBSPHandle BioAPI-HANDLE
        }
```

18.8.1 Общие положения

18.8.1.1 Поле данной таблицы представляет собой локальную связь сессии присоединения, где связь, установленная путем вызова к **BioAPI_BSPAttach**, удерживается локальным приложением к сессии присоединения активного ПБУ либо в локальной, либо в второстепенной конечной точке.

18.8.1.2 Для ПБУ, активного в локальной конечной точке, исходный обработчик ПБУ и локальный обработчик ПБУ создается структурой (см. 16.13). Для ПБУ, активного во второстепенной конечной точке, исходный обработчик ПБУ создается внутри второстепенной конечной точки при получении сообщения запроса ПМО БиоАПИ **bspAttach** от структуры (см. 16.13), а обработчик ПБУ создается структурой после получения соответствующего сообщения ответа ПМО БиоАПИ **bspAttach**, содержащего исходный обработчик ПБУ (см. 16.13).

18.8.1.3 Данная таблица также поддерживает трансляцию между локальным и исходным обработчиками ПБУ.

Примечание – Данный механизм введен из-за того, что обработчик ПБУ может быть уникальным только в границах структуры, которая его создает. Возвращение приложению созданного второстепенной структурой исходного обработчика ПБУ может привести к противоречию с другим исходным обработчиком ПБУ, созданным другими второстепенными структурами или локальной структурой. Для ПБУ, исполняющегося в локальной конечной точке, структура сначала создает исходный обработчик ПБУ без отношения к имеющемуся локальному обработчику ПБУ, а затем определяет локальный обработчик ПБУ, который может отличаться от исходного обработчика ПБУ.

18.8.2 Компоненты

18.8.2.1 Компонент **hostingEndpointIRI** должен содержать ИИР конечной точки ПМО БиоАПИ, которая включает в себя зарегистрированный ПБУ, относящийся к локальной связи сессии присоединения. Конечная точка ПМО БиоАПИ должна быть либо локальной, либо второстепенной конечной точкой.

18.8.2.2 Компонент **bspProductUuid** содержит УУИД продукта ПБУ зарегистрированного ПБУ.

18.8.2.3 Компонент **useBSPAccessUuid** должен определять, предоставило ли локальное приложение УУИД доступа ПБУ (а не УУИД продукта ПБУ) для идентификации зарегистрированного ПБУ при вызове **BioAPI_BSPAttach** (см. 16.13), который вызвал добавление поля.

18.8.2.4 Компонент **originalBSPHandle** должен содержать исходный обработчик ПБУ.

18.8.2.5 Компонент **localBSPHandle** должен содержать локальный обработчик ПБУ.

18.8.2.6 Не должно быть двух полей с одинаковыми значениями компонента **localBSPHandle** и двух полей с одинаковыми значениями компонентов **hostingEndpointIRI** и **originalBSPHandle**.

18.8.3 Удаление поля

18.8.3.1 Данный подпункт применяют только в том случае, если на него имеется ссылка в других пунктах настоящего стандарта, когда поле таблицы **AttachSessionLocalReferences** должно быть удалено.

18.8.3.2 Разрешить *hostingEndpointIRI* выступать в качестве значения компонента **hostingEndpointIRI** и *originalBSPHandle* в качестве значения компонента **originalBSPHandle** того поля, которое должно быть удалено.

18.8.3.3 Для каждого поля таблицы **GUIEventLocalSubscriptions** (см. 18.10), в котором компонент **hostingEndpointIRI** имеет значение *hostingEndpointIRI* и компонент **originalBSPHandle** присутствует и имеет значение *originalBSPHandle*, структура должна выполнить следующие действия в указанном порядке:

а) в случае, если *hostingEndpointIRI* не является ИИР локальной конечной точки, удалить поле таблицы **GUIEventLocalSubscriptions** (выполняя действия, указанные в 18.10.3) без выполнения следующих действий;

b) создать временное абстрактное значение (*unsubscribeFromGUIEventsCallParams*) типа **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** устанавливается из необязательного компонента **guiEventSubscriptionUuid** поля (присутствие и значение);

2) в случае, если необязательный компонент **originalBSPHandle** поля отсутствует, необязательный компонент **bspUuid** из *unsubscribeFromGUIEventsCallParams* должен быть установлен из компонента **bspProductUuid** поля; в противном случае, компонент должен отсутствовать;

3) необязательный компонент **bspHandle** устанавливают из необязательного компонента **originalBSPHandle** поля (присутствие и значение);

4) в случае, если компонент **guiSelectEventHandlerAddress** поля имеет значение, отличающееся от 0, компонент **guiSelectEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливается на 0;

5) в случае, если компонент **guiStateEventHandlerAddress** поля имеет значение, отличающееся от 0, компонент **guiStateEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливается на 0;

6) в случае, если компонент **guiProgressEventHandlerAddress** поля имеет значение, отличающееся от 0, компонент **guiProgressEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен соответствовать используемому и в 16.22.5, перечисление b); в противном случае компонент устанавливается на 0 и

7) компоненты **guiSelectEventHandlerContext**, **guiStateEventHandlerContext**, и **guiProgressEventHandlerContext** устанавливают на 0;

c) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnsubscribeFromGUIEvents** (см. 16.23), в котором параметры вызова функции устанавливают путем преобразования из *unsubscribeFromGUIEventsCallParams* согласно 16.23.6;

d) удалить поле таблицы **GUIEventLocalSubscriptions**, выполняя действия, указанные в 18.10.3.

18.8.3.4 Если *hostingEndpointIRI* является ИИР локальной конечной точки, тогда, для каждого поля таблицы **GUIEventRedirectors** (см. 18.12), в котором компонент **originalBSPHandle** имеет значение *originalBSPHandle*, структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*unredirectCallParams*) типа **UnredirectGUIEvents- RequestParams** (см. 16.29.2), в котором все компоненты должны быть установлены из компонентов поля с такими же именами;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnredirectGUIEvents** (см. 16.29), в котором параметры вызова функции должны быть установлены путем преобразования из *unredirectCallParams* согласно 16.29.5;

c) удалить поле таблицы **GUIEventRedirectors**, выполняя действия, указанные в 18.12.3.

18.8.4 Жизненный цикл

18.8.4.1 Поле может быть добавлено в таблицу **AttachSessionLocalReferences** в следующем случае:

– когда структура получает вызов к функции **BioAPI_BSPAttach** от локального приложения (см. 16.13).

18.8.4.2 Поле таблицы **AttachSessionLocalReferences** может быть удалено в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_BSPDetach** от локального приложения (см. 16.14) и

b) когда поле таблицы **RunningBSPLocalReferences** должно быть удалено (см. 18.5.4.2).

18.9 Концептуальная таблица AttachSessionRemoteReferences

Данная концептуальная таблица представлена во всех второстепенных конечных точках и определено в АСН.1 следующим образом :

```
AttachSessionRemoteReferences ::= SET OF
        reference AttachSessionRemoteReference

AttachSessionRemoteReference ::= SEQUENCE {
        referrerEndpointIRI EndpointIRI,
        bspProductUuid BioAPI-UUID,
        originalBSPHandle BioAPI-HANDLE
    }
```

18.9.1 Общие положения

Поле данной таблицы представляет собой удаленную связь сессии присоединения, где связь, установленная сообщением запроса ПМО БиоАПИ **bspLoad**, удерживается главной конечной точкой к сессии присоединения активного ПБУ в локальной конечной точке.

18.9.2 Компоненты

18.9.2.1 Компонент **referrerEndpointIRI** должен содержать ИИР конечной точки главной конечной точки, которая удерживает удаленную связь сессии присоединения и не должна быть локальной конечной точкой.

18.9.2.2 Компонент **bspProductUuid** должен содержать УУИД продукта ПБУ зарегистрированного ПБУ.

18.9.2.3 Компонент **originalBSPHandle** должен содержать обработчик сессии присоединения.

18.9.2.4 Не должно быть двух полей с одинаковыми значениями компонента **originalBSPHandle**.

18.9.3 Удаление поля

18.9.3.1 Данный пункт применяют только в том случае, если на него имеются ссылки в других пунктах настоящего стандарта, когда поле таблицы **AttachSessionRemoteReferences** должно быть удалено.

18.9.3.2 Разрешить *originalBSPHandle* выступать в качестве значения компонента **originalBSPHandle** того поля, которое удаляется.

18.9.3.3 Для каждого поля таблицы **GUIEventRemoteSubscriptions** (см. 18.11), в котором компонент **originalBSPHandle** присутствует и имеет значение *originalBSPHandle*, структура должна выполнить следующие действия в указанном порядке:

а) создать временное абстрактное значение (*unsubscribeFromGUIEventsCallParams*) типа **UnsubscribeFromGUIEventsCallParams** (см. 16.23.3), в котором:

1) необязательный компонент **guiEventSubscriptionUuid** должен быть установлен из необязательного компонента **guiEventSubscriptionUuid** поля (присутствие и значение);

2) в случае, если необязательный компонент **originalBSPHandle** поля отсутствует, необязательный компонент **bspUuid** *unsubscribeFromGUIEventsCallParams* должен быть установлен из

компонента **bspProductUuid** поля; в противном случае компонент должен отсутствовать;

3) необязательный компонент **bspHandle** должен быть установлен из необязательного компонента **originalBSPHandle** поля (присутствие и значение);

4) в случае, если компонент **guiSelectEventSubscribed** поля имеет значение **TRUE**, компонент **guiSelectEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен быть аналогичен используемому и в 16.22.5, перечисление b); в противном случае компонент должен быть установлен на 0;

5) Если компонент **guiStateEventSubscribed** поля имеет значение **TRUE**, компонент **guiStateEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен быть аналогичен используемому 16.22.5, перечисление b); в противном случае компонент должен быть установлен на 0;

6) в случае, если компонент **guiProgressEventSubscribed** поля имеет значение **TRUE**, компонент **guiProgressEventHandlerAddress** из *unsubscribeFromGUIEventsCallParams* должен быть установлен в определенный реализацией адрес памяти, отличающийся от 0, который должен быть аналогичен используемому в 16.22.5, перечисление b); в противном случае компонент должен быть установлен на 0 и

7) компоненты **guiSelectEventHandlerContext**, **guiStateEventHandlerContext** и **guiProgressEventHandlerContext** устанавливают на 0;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnsubscribeFromGUIEvents** (см. 16.23), в котором параметры

вызова функции устанавливают путем преобразования из *unsubscribeFromGUIEventsCallParams* согласно 16.23.6;

c) удалить поле таблицы **GUIEventRemoteSubscriptions**, выполняя действия, указанные в 18.11.3;

18.9.3.4 Для каждого поля таблицы **GUIEventRedirectors** (см. 18.12), в котором компонент **originalBSPHandle** имеет значение *originalBSPHandle*, структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*unredirectCallParams*) типа **UnredirectGUIEvents- RequestParams** (см. 16.29.2), в котором все компоненты должны быть установлены из компонентов поля с такими же именами;

b) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции **BioAPI_UnredirectGUIEvents** (см. 16.29), в котором параметры вызова функции устанавливают путем преобразования из *unredirectCallParams* согласно 16.29.5;

c) удалить поле таблицы **GUIEventRedirectors**, выполняя действия, указанные в 18.12.3.

18.9.4 Жизненный цикл

18.9.4.1 Поле может быть добавлено в таблицу **AttachSessionRemoteReferences** в следующем случае:

– когда структура получает сообщение запроса ПМО БиоАПИ **bspAttach** от главной конечной точки (см. 16.13.2).

18.9.4.2 Поле таблицы **AttachSessionRemoteReferences** может быть удалено в следующих случаях:

a) когда структура получает сообщение запроса ПМО БиоАПИ **bspDetach** от главной конечной точки (см. 16.14.2) и

b) когда поле таблицы **RunningBSPRemoteReferences** должно быть удалено (см. 16.6.4.2).

18.10 Концептуальная таблица GUIEventLocalSubscriptions

Данная концептуальная таблица представлена во всех конечных точках ПМО БиоАПИ и определена в АСН.1 следующим образом:

```
GUIEventLocalSubscriptions ::= SET OF
        subscription GUIEventLocalSubscription

GUIEventLocalSubscription ::= SEQUENCE {
                guiEventSubscriptionUuid          BioAPI-UUID OPTIONAL,
                hostingEndpointIRI                EndpointIRI,
                bspProductUuid                    BioAPI-UUID,
                useBSPAccessUuid                  BOOLEAN,
                originalBSPHandle                 BioAPI-HANDLE OPTIONAL,
                guiSelectEventHandlerAddress          MemoryAddress,
                guiSelectEventHandlerContext          MemoryAddress,
                guiStateEventHandlerAddress           MemoryAddress,
                guiStateEventHandlerContext           MemoryAddress,
                guiProgressEventHandlerAddress    MemoryAddress,
                guiProgressEventHandlerContext    MemoryAddress
        }
```

18.10.1 Общие положения

18.10.1.1 Поле данной таблицы представляет собой локальную подписку на операции ГИП, которая является обязательством структуры по совершению обратных вызовов уведомлений об операциях ГИП в локальное приложение по определенным уведомлениям об операциях ГИП (либо обратные вызовы уведомления об операциях ГИП от ПБУ, либо сообщения уведомления ПМО БиоАПИ об операциях ГИП от второстепенной локальной точки) и входящим запросам (либо вызов запроса уведомления об операциях ГИП от локального приложения, либо сообщения уведомления ПМО БиоАПИ об операциях ГИП от главной конечной точки).

18.10.1.2 Входящий вызов к **BioAPI_SubscribeToGUIEvents** может создать новое поле локальной подписки операций ГИП даже в том случае, если параметры вызова аналогичны параметрам предыдущего входящего вызова к **BioAPI_SubscribeToGUIEvents**. Входящий вызов к

BioAPI_UnsubscribeFromGUIEvents, параметры которой совпадают с существующим полем локальной подписки операций ГИП, может удалить такое поле. В случае множественных совпадений одно любое из совпадающих полей может быть удалено.

18.10.2 Компоненты

18.10.2.1 Необязательный компонент **guiEventSubscriptionUuid** (в случае его наличия) должен содержать УУИД подписки на операции ГИП. Отсутствие данного компонента свидетельствует об первичной подписке на операции ГИП.

18.10.2.2 Компонент **hostingEndpointIRI** должен содержать ИИР конечной точки конечной точки ПМО БиоАПИ (либо локальной конечной точки либо второстепенной конечной точки) и означает, что подписка ограничена операциями ГИП исполняющего в этой конечной точке ПМО БиоАПИ ПБУ.

18.10.2.3 Компонент **bspProductUuid** должен содержать УУИД продукта ПБУ и означает, что подписка ограничена операциями ГИП, относящимися к ПБУ с таким УУИД продукта ПБУ.

18.10.2.4 Компонент **useBSPAccessUuid** должен определять, предоставило ли локальное приложение УУИД доступа ПБУ (а не УУИД продукта ПБУ) для идентификации ПБУ при вызове **BioAPI_SubscribeToGUIEvents** (см. 16.22), который вызвал добавление поля.

Примечание – Последующий вызов **BioAPI_UnsubscribeFromGUIEvents** (см. 15.23) должен предоставить такой же УУИД (либо УУИД доступа ПБУ либо УУИД продукта ПБУ) для совпадения с данной локальной подпиской операций ГИП. Обратные вызовы уведомлений операций ГИП, которые создаются с использованием такой локальной подписки операций ГИП, будут передавать УУИД, аналогичный входящему в обработчик операций ГИП локального приложения.

18.10.2.5 Необязательный компонент **originalBSPHandle** (в случае его наличия) должен содержать исходный обработчик ПБУ и указывает, что

подписка ограничена операциями ГИП, несущих такой обработчик ПБУ. Отсутствие данного компонента свидетельствует о том, что подписка не ограничена определенной сессией присоединения.

18.10.2.6 Компонент **guiSelectEventHandlerAddress** должен содержать адрес обратного вызова обработчика операции выбора ГИП локального приложения. Значение 0 означает, что об операциях выбора ГИП подписчик не уведомляется.

18.10.2.7 Компонент **guiSelectEventHandlerContext** должен содержать адрес контекста, который должен передаваться как входящие данные в обработчик выбора ГИП.

18.10.2.8 Компонент **guiStateEventHandlerAddress** должен содержать адрес обратного вызова обработчика операции состояния ГИП локального приложения. Значение 0 означает, что об операциях статуса ГИП подписчик не уведомляется.

18.10.2.9 Компонент **guiStateEventHandlerContext** должен содержать адрес контекста, который должен передаваться как входящие данные в обработчик статуса ГИП.

18.10.2.10 Компонент **guiProgressEventHandlerAddress** должен содержать адрес обратного вызова обработчика операции прогресса ГИП локального приложения. Значение 0 означает, что об операциях прогресса ГИП подписчик не уведомляется. Один или более компонентов **guiSelectEventHandlerAddress**, **guiStateEventHandlerAddress** и **guiProgressEventHandlerAddress** должен иметь значение, отличающееся от 0.

18.10.2.11 Компонент **guiProgressEventHandlerContext** должен содержать адрес контекста, который должен передаваться как входящие данные в обработчик прогресса ГИП.

18.10.2.12 Допускается наличие множества полей с одинаковыми значениями одного или более (или всех) их компонентов.

18.10.3 Удаление поля

18.10.3.1 Данный подпункт применяют только в том случае, если на него имеется ссылка в других пунктах настоящего стандарта, когда поле таблицы **GUIEventLocalSubscriptions** должно быть удалено.

18.10.3.2 Дополнительные действия не требуются.

18.10.4 Жизненный цикл

18.10.4.1 Поле может быть добавлено в таблицу **GUIEventLocalSubscriptions** в следующем случае:

– когда структура получает вызов к функции **BioAPI_SubscribeToGUIEvents** от локального приложения (см. 15.22).

18.10.4.2 Поле таблицы **GUIEventLocalSubscriptions** может быть удалено в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_UnsubscribeFromGUIEvents** от локального приложения (см. 16.23);

b) когда поле таблицы **VisibleBSPRegistrations** должно быть удалено (см. 18.3.4.2) и

c) когда поле таблицы **AttachSessionLocalReferences** должно быть удалено (см. 18.8.4.2).

18.11 Концептуальная таблица GUIEventRemoteSubscriptions

Данная концептуальная таблица представлена во всех второстепенных конечных точках и определена в ACH.1 следующим образом :

```
GUIEventRemoteSubscriptions ::= SET OF
        subscription GUIEventRemoteSubscription

GUIEventRemoteSubscription ::= SEQUENCE {
        subscriberEndpointIRI EndpointIRI,
        guiEventSubscriptionUuid BioAPI-UUID OPTIONAL,
        bspProductUuid BioAPI-UUID,
        originalBSPHandle BioAPI-HANDLE OPTIONAL,
        guiSelectEventSubscribed BOOLEAN,
```

guiStateEventSubscribed BOOLEAN,
guiProgressEventSubscribed BOOLEAN
}

18.11.1 Общие положения

18.11.1.1 Поле данной таблицы представляет собой удаленную подписку на операции ГИП, которая является обязательством структуры по отправлению сообщений уведомления ПМО БиоАПИ об операциях ГИП в главную локальную точку по определенным входящим уведомлениям об операциях ГИП (обратные вызовы уведомления об операциях ГИП от ПБУ) и входящим запросам (либо вызов запроса уведомления об операциях ГИП от локального приложения, либо сообщения уведомления ПМО БиоАПИ об операциях ГИП от главной конечной точки).

18.11.1.2 Входящее сообщение запроса ПМО БиоАПИ **subscribeToGUIEvents** может создать новое поле удаленной подписки на операции ГИП даже в том случае, если значение параметра сообщения запроса ПМО БиоАПИ является аналогичным значению параметра предыдущего входящего сообщения запроса ПМО БиоАПИ **subscribeToGUIEvents**. Входящее сообщение запроса ПМО БиоАПИ **unsubscribeFromGUIEvents**, компоненты которого совпадают с существующим полем удаленной подписки операций ГИП, может удалить такое поле. В случае множественных совпадений любое поле из совпадающих может быть удалено.

18.11.2 Компоненты

18.11.2.1 Компонент **subscriberEndpointIRI** должен содержать ИИР конечной точки главной конечной точки, которая подписана на операции ГИП.

18.11.2.2 Необязательный компонент **guiEventSubscriptionUuid** (в случае его наличия) должен содержать УУИД подписки на операции ГИП. Отсутствие данного компонента свидетельствует об первичной подписке операций ГИП.

18.11.2.3 Компонент **bspProductUuid** должен содержать УУИД продукта ПБУ и указывает, что подписка ограничена операциями ГИП, связанными с ПБУ с этим УУИД продукта ПБУ.

18.11.2.4 Необязательный компонент **originalBSPHandle** (в случае его наличия) должен содержать исходный обработчик ПБУ и указывает, что подписка ограничена операциями ГИП, которые связаны с сессией присоединения, идентифицированной этим обработчиком ПБУ. Отсутствие данного компонента свидетельствует о том, что подписка не ограничена определенной сессией присоединения. Данный компонент должен присутствовать только в том случае, если необязательный компонент **guiEventSubscriptionUuid** отсутствует.

18.11.2.5 Компонент **guiSelectEventSubscribed** определяет необходимость уведомления подписчика об операциях выбора ГИП.

18.11.2.6 Компонент **guiStateEventSubscribed** определяет необходимость уведомления подписчика об операциях состояния ГИП.

18.11.2.7 Компонент **guiProgressEventSubscribed** определяет необходимость уведомления подписчика об операциях прогресса ГИП. Один или более из компонентов **guiSelectEventSubscribed**, **guiStateEventSubscribed**, и **guiProgressEventSubscribed** должны иметь значение **TRUE**.

18.11.2.8 Допускается наличие множества полей с одинаковыми значениями одного или более (или всех) их компонентов.

18.11.3 Удаление поля

18.11.3.1 Данный пункт применяют только в том случае, если на него 8имеются ссылки в других пунктах настоящего стандарта, когда поле таблицы **GUIEventRemoteSubscriptions** должно быть удалено.

18.11.3.2 Дополнительные действия не требуются.

18.11.4 Жизненный цикл

18.11.4.1 Поле может быть добавлено в таблицу **GUIEventRemoteSubscriptions** в следующем случае:

– когда структура получает сообщение запроса ПМО БиоАПИ **subscribeToGUIEvents** от главной конечной точки (см. 16.22.2).

18.11.4.2 Поле таблицы **GUIEventRemoteSubscriptions** может быть удалено в следующих случаях:

a) когда структура получает сообщение запроса ПМО БиоАПИ **unsubscribeFromGUIEvents** от главной конечной точки (см. 16.23.2);

b) когда поле таблицы **MasterEndpoints** должно быть удалено (см. 18.1.4.2);

c) когда поле таблицы **VisibleBSPRegistrations** должно быть удалено (см. 18.3.4.2) и

d) когда поле таблицы **AttachSessionRemoteReferences** должно быть удалено (см. 18.9.4.2).

18.12 Концептуальная таблица GUIEventRedirectors

Данная концептуальная таблица представлена во всех конечных точках ПМО БиоАПИ и определена в АСН.1 следующим образом:

```
GUIEventRedirectors ::= SET OF
        redirector GUIEventRedirector

GUIEventRedirector ::= SEQUENCE {
        referrerEndpointIRI EndpointIRI,
        bspProductUuid BioAPI-UUID,
        originalBSPHandle BioAPI-HANDLE,
        subscriberEndpointIRI EndpointIRI,
        guiEventSubscriptionUuid BioAPI-UUID,
        guiSelectEventRedirected BOOLEAN,
        guiStateEventRedirected BOOLEAN,
        guiProgressEventRedirected BOOLEAN
    }
```

18.12.1 Общие положения

18.12.1.1 Поле данной таблицы представляет собой редиректор операций ГИП и является обязательством структуры по обработке обратных вызовов уведомления об операция ГИП, полученных от исполняющегося в локальном приложении ПБУ (для заданной сессии присоединения), как если бы они были входящими запросами уведомления об операциях ГИП с определенными параметрами.

18.12.1.2 Входящий вызов к **BioAPI_RedirectGUIEvents** может создать новое поле редиректора операций ГИП даже в том случае, если параметры вызова являются аналогичными параметрам предыдущего вызова к **BioAPI_RedirectGUIEvents**. Входящий вызов к **BioAPI_UnredirectGUIEvents**, компоненты которого совпадают с существующим полем редиректора операций ГИП, может удалить такое поле. В случае множественных совпадений, одно из совпадающих полей может быть удалено.

18.12.1.3 Входящее сообщение запроса ПМО БиоАПИ **redirectGUIEvents** может создать новое поле даже в том случае, если значение параметра сообщения запроса ПМО БиоАПИ является аналогичным значению параметра предыдущего входящего сообщения запроса ПМО БиоАПИ **redirectGUIEvents**. Входящее сообщение запроса ПМО БиоАПИ **unredirectGUIEvents**, компоненты которого аналогичны существующему полю редиректора операций ГИП, может удалить это поле. В случае множественных совпадений одно из совпадающих полей может быть удалено.

18.12.2 Компоненты

18.12.2.1 Компонент **referrerEndpointIRI** должен содержать ИИР конечной точки конечной точки ПМО БиоАПИ, которая удерживает связь (либо локальную связь сессии присоединения, либо удаленную связь сессии присоединения) к сессии присоединения в редиректоре операций ГИП. Либо это может главная конечная точка локальной конечной точки.

18.12.2.2 Компонент **bspProductUuid** должен содержать УУИД продукта ПБУ, содержащегося в редиректоре операций ГИП.

18.12.2.3 Компонент **originalBSPHandle** должен содержать исходный обработчик ПБУ и означает, что редиректор операций ГИП ограничен операциями ГИП, которые связаны с сессией присоединения, идентифицированной таким обработчиком ПБУ.

18.12.2.4 Компонент **subscriberEndpointIRI** должен содержать ИИР конечной точки конечной точки ПМО БиоАПИ, которая должна получать перенаправленные уведомления об операциях ГИП. Либо это может главная конечная точка локальной конечной точки.

18.12.2.5 Компонент **guiEventSubscriptionUuid** должен содержать УУИД подписки на операции ГИП, который должен быть установлен для компонента **guiEventSubscriptionUuid** из перенаправленных уведомлений об операциях ГИП.

18.12.2.6 Компонент **guiSelectEventRedirected** определяет необходимость перенаправления подписчику операций выбора ГИП.

18.12.2.7 Компонент **guiStateEventRedirected** определяет необходимость перенаправления подписчику операций состояния ГИП.

18.12.2.8 Компонент **guiProgressEventRedirected** определяет необходимость перенаправления подписчику операций прогресса ГИП. Один или более компонентов **guiSelectEventRedirected**, **guiStateEventRedirected** и **guiProgressEventRedirected** должны иметь значение **TRUE**.

18.12.2.9 Допускается наличие множества полей с одинаковыми значениями одного или более (или всех) их компонентов.

18.12.3 Удаление поля

18.12.3.1 Данный пункт применяют только в том случае, если на него имеется ссылка в других пунктах настоящего стандарта, когда поле таблицы **GUIEventRedirectors** должно быть удалено.

18.12.3.2 Дополнительные действия не требуются.

18.12.4 Жизненный цикл

18.12.4.1 Поле может быть добавлено в таблицу **GUIEventRedirectors** в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_RedirectGUIEvents** от локального приложения (см. 16.28);и

b) когда структура получает сообщение запроса ПМО БиоАПИ **redirectGUIEvents** от главной конечной точки (см. 16.28.2).

18.12.4.2 Поле таблицы **GUIEventRedirectors** может быть удалено в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_UnredirectGUIEvents** от локального приложения (см. 16.29);

b) когда структура получает сообщение запроса ПМО БиоАПИ **unredirectGUIEvents** от главной конечной точки (см. 16.29.2);

c) когда поле таблицы **MasterEndpoints** должно быть удалено (см. 18.1.4.2);

d) когда поле таблицы **AttachSessionLocalReferences** должно быть удалено (см. 18.8.4.2) и

e) когда поле таблицы **AttachSessionRemoteReferences** должно быть удалено (см. 18.9.4.2).

18.13 Концептуальная таблица ApplicationOwnedMemoryBlocks

Данная концептуальная таблица представлена во всех конечных точках ПМО БиоАПИ и определена в АСН.1 следующим образом:

```
ApplicationOwnedMemoryBlocks ::= SET OF
        memoryBlock ApplicationOwnedMemoryBlock

ApplicationOwnedMemoryBlock ::= SEQUENCE {
        address MemoryAddress
    }
```

18.13.1 Общие положения

18.13.1.1 Поле данной таблицы предоставляет собой блок памяти, образованный структурой, но принадлежащий локальному приложению.

18.13.1.2 Данная таблица содержит запись образования памяти, которое выполнялось структурой таким образом, что блоки памяти могут быть освобождены либо входящим вызовом к **BioAPI_Free** (см. 16.58), либо входящим вызовом к **BioAPI_Terminate** (см. 16.3) от локального приложения.

18.13.2 Компоненты

18.13.2.1 Компонент **address** должен содержать адрес блока памяти, который представлен полем.

18.13.2.2 Не должно быть двух или более полей с одинаковыми значениями компонента **address**.

18.13.3 Жизненный цикл

18.13.3.1 Поле может быть добавлено в таблицу **ApplicationOwnedMemoryBlocks** в следующем случае:

– когда образованные переменная или массив создаются во время обработки входящего вызова функции БиоАПИ от локального приложения (см. 13.13).

18.13.3.2 Поле таблицы **ApplicationOwnedMemoryBlocks** может быть удалено в следующих случаях:

a) когда структура получает вызов к функции **BioAPI_Terminate** от локального приложения (см. 16.3) и

b) когда структура получает вызов к функции **BioAPI_Free** от локального приложения (см. 16.58).

19 Преобразования между переменной указателя Си и соответствующим компонентом ACH.1 (1)

19.1 Данный раздел применяется только в том случае, если на него имеется прямая ссылка в других пунктах настоящего стандарта. В разделе определено преобразование:

a) между переменной указателя Си, которая является членом большей структуры, и компонентом ACH.1, который соответствует типу Си указанной структуры, либо

b) между переменной указателя Си, которая является входящим параметром функции, и компонентом типа ACH.1.

19.2 Вызов *Type* к типу выделенной переменной выполняют согласно определению переменной указателя Си.

19.3 Преобразование из переменной указателя Си в компонент ACH.1 выполняют следующим образом:

a) в случае, если переменная указателя Си имеет значение **NULL** и компонент ACH.1 не имеет значение **OPTIONAL**, компонент ACH.1 должен отсутствовать;

b) в случае, если переменная указателя Си имеет значение **NULL** и компонент ACH.1 не имеет значение **OPTIONAL**, значение Си не должна быть преобразована и применяют раздел 33;

c) в случае, если переменная указателя Си имеет значение, отличающееся от **NULL**, переменная типа *Type*, выделенная переменной указателя Си, должна быть преобразована в компонент ACH.1, согласно подразделу, на который ссылаются в обращении данного раздела.

19.4 Преобразование из компонента ACH.1 в переменную указателя Си выполняют следующим образом:

a) в случае, если компонент ACH.1 является **OPTIONAL** и отсутствует, переменная указателя Си должна быть установлена на **NULL**;

b) в случае, если компонент ACH.1 присутствует, переменная указателя Си должна быть установлена в адрес новообразованной переменной типа *Type*, а компонент ACH.1 должен быть преобразован в переменную согласно подразделу, на который ссылаются в обращении данного раздела.

20 Преобразования между переменной указателя Си и соответствующим компонентом ACH.1 (2)

20.1 Данный раздел применяют в том случае, если на него имеется ссылка в других разделах настоящего стандарта. В разделе определено преобразование между переменной указателя Си, которая является выходным параметром функции, и компонентом типа ACH.1.

20.2 Вызов *Type* к типу выделенной переменной, выполняют аналогично определению переменной указателя Си.

20.3 Преобразование из переменной указателя Си в компонент ACH.1 выполняют следующим образом:

a) в случае, если переменная указателя Си имеет значение **NULL**, компонент ACH.1 должен отсутствовать.

Примечание – Данные случай может возникнуть, когда компонентом ACH.1 является **OPTIONAL**;

b) в случае, если переменная указателя Си имеет значение, отличающееся от **NULL**, компонент ACH.1 должен присутствовать и переменная типа *Type*, выделенная переменной указателя Си, преобразуется в такой компонент согласно подразделу, на который ссылаются в обращении данного раздела.

20.4 Преобразование из компонента ACH.1 в переменную указателя Си выполняют следующим образом:

a) в случае, если переменная указателя Си имеет значение **NULL**, никакие действия не требуются.

Примечание 1 – Если компонент ACH.1 присутствует, он будет проигнорирован (такая ситуация не возникнет с сообщением ответа или подтверждения ПМО БиоАПИ, которые получены от соответствующей точки ПМО БиоАПИ);

b) в случае, если компонентом ACH.1 является **OPTIONAL** и отсутствует, никакие действия не требуются.

Примечание 2 – Если переменная указателя Си имеет значение, отличающееся от **NULL**, переменная типа *Type*, выделенная переменной указателя Си сохранит свое текущее значение (такая ситуация не возникает с сообщением ответа или подтверждения ПМО БиоАПИ, которые получены от соответствующей точки ПМО БиоАПИ);

c) в случае, если переменная указателя Си имеет значение, отличающееся от **NULL**, а компонент ACH.1 присутствует, компонент ACH.1 преобразуется в переменную типа *Type*, выделенную переменной указателя Си согласно подразделу, на который ссылаются в обращении данного раздела.

21 Преобразования между переменной указателя Си и соответствующим компонентом ACH.1 (3)

21.1 Данный раздел применяют в том случае, если на него имеется ссылка в других разделах настоящего стандарта. В разделе определено преобразование между переменной указателя Си, которая является выходным параметром функции, и компонентом типа ACH.1.

Примечание – Соответствующий компонент ACH.1 всегда является типом **BOOLEAN** и распознает, имеет ли переменная указателя Си значение **NULL**. Все булевые компоненты ACH.1 имеют имена, начинающиеся с «**no-**», с целью подчеркнуть смысл того, что соответствующий исходный параметр не требуется.

21.2 Вызов *Type* к типу выделенной переменной выполняют аналогично определению переменной указателя Си.

21.3 Преобразование из переменной указателя Си в компонент ACH.1 выполняют следующим образом: в случае, если переменная указателя Си имеет

значение **NULL**, компонент АСН.1 устанавливается на **TRUE**. В противном случае компонент АСН.1 устанавливают на **FALSE**.

Примечание 1 – Если переменная указателя Си имеет значение, отличающееся от **NULL**, значение переменной типа *Type*, выделенное переменной указателя Си, будет проигнорировано.

21.4 Преобразование из компонента АСН.1 в переменную указателя Си выполняют следующим образом: Если компонент АСН.1 имеет значение **TRUE**, переменная указателя Си должна быть установлена на **NULL**. В противном случае переменная указателя Си должна быть установлена в адрес новообразованной переменной типа *Type*, а глубинный блок памяти (число октетов соответствует размеру переменной) должен быть заполнен нулями.

22 Инициализация и проверка переменной указателя Си, не имеющей соответствующего компонентв АСН.1

22.1 Данный раздел применяют только в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта. В разделе определена инициализация и проверка переменной указателя СИ, которая является выходным параметром функции и не имеет соответствующего компонента в данном типе АСН.1.

22.2 При преобразовании из параметров функции БиоАПИ в тип АСН.1, должна быть выполнена следующая проверка: в случае, если переменная указателя Си имеет значение **NULL**, значение Си не должно быть преобразовано и применяют раздел 33.

22.3 При преобразовании типа АСН.1 в параметры функции БиоАПИ, переменная указателя Си переменная указателя Си должна быть установлена в адрес вновь образованной переменной типа *Type*, а глубинный блок памяти (число октетов соответствует размеру переменной) должно быть заполнено нулями.

Примечание – В некоторых случаях, вновь образованная переменная является указателем. В таких случаях размер глубинного блока памяти будет равен размеру указателя, а указатель будет инициализирован на **NULL**.

23 Определение главной конечной точки и УУИД продукта ПБУ из УУИД ПБУ

23.1 Данный раздел применяют только в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для определения главной конечной точки ПБУ входящего вызова к функции БиоАПИ, который содержит как и параметр **BSPUuid** типа **const BioAPI_UUID***, так и УУИД продукта ПБУ

23.2 Структура должна выполнить следующие действия в указанном порядке:

a) преобразовать параметр **BSPUuid** во вновь созданное временное абстрактное значение (***bspUuid***) типа **BioAPI-UUID** согласно разделу 19 совместно с 16.58;

b) проверить таблицу **VisibleBSPRegistrations** (см. 18.3) на наличие двух полей, в которых либо компонент bspAccessUuid, либо компонент **bspProductUuid** поля имеет значение bspUuid;

c) в случае, если соответствующее поле отсутствует, удостовериться, что главная конечная точка не может быть определена;

Примечание 1 – УУИД, предоставленный локальным приложением неизвестен, так как он не является УУИД продукта ПБУ или УУИД доступа ПБУ.

d) в случае, если существует только одно совпадающее поле, удостовериться, что главная конечная точка является конечной точкой ПМО БиоАПИ, которая идентифицирована компонентом **hostingEndpointIRI** поля, а УУИД продукта ПБУ является значением компонента **bspProductUuid** этого поля;

e) в случае, если существует два или более совпадающих полей, удостовериться, что главная конечная точка не может быть определена.

Примечание 2 – Вышеуказанное может произойти Если УУИД, предоставленный локальным приложением, является УУИД продукта ПБУ и идентифицирует ПБУ, как загружаемый или исполняющийся в нескольких главных конечных точках. Локальное приложение должно предоставить УУИД доступа ПБУ в вызове функции.

24 Определение главной конечной точки и исходного обработчика ПБУ из локального обработчика ПБУ

24.1 Данный раздел применяют только в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для определения главной конечной точки ПБУ входящего вызова к функции БиоАПИ, который имеет как параметр **BSPHandle** типа **BioAPI_HANDLE** (или **const BioAPI_HANDLE***), так и исходный обработчика ПБУ.

24.2 Структура должна выполнить следующие действия в указанном порядке:

a) преобразовать параметр **BSPHandle** во вновь созданное временное абстрактное значение (*localBSPHandle*) типа **BioAPI-HANDLE** согласно 16.42 (или согласно разделу 19 совместно с 16.42 в случае, если типом параметра является **const BioAPI_HANDLE***);

b) проверить таблицу **AttachSessionLocalReferences** (см. 18.8) на наличие поля, в котором компонент **localBSPHandle** имеет значение *localBSPHandle*.

Примечание 1 – Может быть только одно соответствующее поле (см. 16.13);

c) в случае, если соответствующее поле отсутствует, удостовериться, что главная конечная точка не может быть определена.

Примечание 2 – Обработчик ПБУ, предоставленный локальным приложением, неизвестен;

d) в случае, если существует только одно совпадающее поле (*localReference*), удостовериться, что главной конечной точкой является

та, которая идентифицирована значением компонента **hostingEndpointIRI** из *localReference*, а исходный обработчик ПБУ является значением компонента **originalBSPHandle** из *localReference*.

25 Преобразования УУИД ПБУ

25.1 Данный раздел применяют только в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для преобразования между параметром функции СИ типа **const BioAPI_UUID*** (содержащим либо УУИД продукта ПБУ, либо УУИД доступа ПБУ) и компонентом АСН.1 типа **BioAPI-UUID** (содержащим УУИД продукта ПБУ).

25.2 Преобразование из параметра Си в компонент АСН.1 выполняют путем установки абстрактного значения АСН.1 в УУИД продукта ПБУ, который должен быть определен согласно разделу 23.

25.3 Преобразование из компонента АСН.1 в параметр Си выполняют согласно 16.58.

26 Преобразования обработчиков ПБУ

26.1 Данный раздел применяют только в том случае, если на него имеется ссылка в других разделах настоящего стандарта для преобразования между параметром функции СИ типа **BioAPI_HANDLE** (содержащим локальный обработчик ПБУ) и компонентом АСН.1 типа **BioAPI-HANDLE** (содержащим исходный обработчик ПБУ).

26.2 Преобразование из параметра Си в компонент АСН.1 выполняют путем установки абстрактного значения АСН.1 в исходный обработчик ПБУ, который должен быть определен согласно разделу 24.

26.3 Преобразование из компонента АСН.1 в параметр Си выполняют согласно 16.42.

27 Обработка входящего вызова функции путем обмена с второстепенной конечной точкой двумя сообщениями запроса/ответа ПМО БиоАПИ

27.1 Данный раздел применяют только в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для обработки входящего вызова функции путем отправления сообщения запроса ПМО БиоАПИ во второстепенную конечную точку и получения соответствующего сообщения ответа ПМО БиоАПИ от нее.

27.2 Структура должна выполнить следующие действия в указанном порядке:

a) создать временное абстрактное значение (*outgoingRequestParams*) параметра типа АСН.1 сообщения запроса ПМО БиоАПИ, которое указано в обращении данного раздела, путем преобразования из параметров вызова функции с использованием подпункта, который указывается в обращении данного раздела;

b) создать и отправить сообщение запроса ПМО БиоАПИ (см. 13.2) типа сообщений ПМО БиоАПИ, которое указано в обращении данного раздела, с ИИР второстепенной конечной точки, установленным на ИИР конечной точки второстепенной конечной точки, и значением параметром в *outgoingRequestParams*;

c) принять соответствующее сообщение ответа ПМО БиоАПИ (см. 13.6);

d) в случае, если возвращенное значение сообщения ответа ПМО БиоАПИ не равно 0, возвратить это значение локальному приложению без выполнения следующих действий;

e) установить исходящие вызова функции путем преобразования значения параметра сообщения в ответ ПМО БиоАПИ, выполняя действия, указанные в подраздел, который указывается в обращении данного раздела;

f) возвратить значение 0 локальному приложению.

28 Обработка входящего сообщения запроса ПМО БиоАПИ путем внутреннего вызова функции БиоАПИ

28.1 Данный раздел применяют только в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для обработки входящего сообщения запроса ПМО БиоАПИ от заданной главной конечной точки путем совершения внутреннего вызова функции БиоАПИ (см. 13.10) и последующего отправления соответствующего сообщения ответа ПМО БиоАПИ.

28.2 Структура должна выполнить следующие действия в указанном порядке:

a) выполнить внутренний вызов функции БиоАПИ (см. 13.10) к функции БиоАПИ, которая указана в обращении данного раздела, в котором параметры вызова функции должны быть установлены путем преобразования из параметра типа ACH.1 сообщения запроса ПМО БиоАПИ, выполняя действия, указанные в подраздел, который указывается в обращении данного раздела.

b) в случае, если возвращенное значение внутреннего вызова не равно 0, создать и отправить соответствующее сообщение ответа ПМО БиоАПИ (см. 13.3) с возвращаемым значением, установленным на это значение без выполнения следующих действий;

c) создать временное абстрактное значение (*incomingResponseParams*) параметра типа ACH.1 сообщения ответа ПМО БиоАПИ (которое указывается в обращении данного раздела) путем преобразования из исходящих параметров внутреннего вызова, выполняя действия, указанные в подраздел, который указан в обращении данного раздела;

d) создать и отправить соответствующее сообщение ответа ПМО БиоАПИ (см. 13.3) со значением параметра, установленным на *incomingResponseParams*, и возвращаемым значением, установленным на 0.

29 Предоставление ни одному или нескольким подписчикам информации о модуле операций

29.1 Данный раздел применяют в том случае, если на него имеется ссылка в других разделах настоящего стандарта, для предоставлении информации о модуле операций, который основан на значении (*eventInfo*) типа **UnitEventInfo** ни одному или нескольким подписчикам, которыми являются:

a) ни один или несколько обработчиков операций локального приложения или

b) ни одна или несколько главных конечных точек

или оба этих случая.

29.2 Структура должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **UnitEventNotificationDisablers** (см. 18.7) на наличие поля, в котором:

1) компонент **referrerEndpointIRI** установлен на ИИР локальной конечной точки;

2) компонент **bspProductUuid** имеет такое же значение, как и компонент **bspProductUuid** из *eventInfo* и

3) компонент **unitEventTypes** имеет значение, означающее, что тип операции идентификации *eventInfo* отключен;

b) в случае, если существует соответствующее поле, не выполнять следующие действия (продолжить со следующего подраздела);

c) проверить таблицу **RunningBSPLocalReferences** (см. 18.5) на наличие всех полей, в которых:

1) компоненты **hostingEndpointIRI** и **bspProductUuid** имеют значения, аналогичные значениям компонентов *eventInfo* с теми же именами и

2) компонент **unitEventHandlerAddress** имеет значение, отличающееся от 0;

d) для каждого соответствующего поля (*localSubscription*) в любом порядке должно быть выполнено следующее: создано временное абстрактное значение (*outgoingCallbackParams*) типа **UnitEventHandlerCallbackParams** (см. 18.1.4), в котором:

1) компоненты **unitEventHandlerAddress** и **unitEventHandlerContext** устанавлены из компонентов *localSubscription* с теми же именами;

2) в случае, если компонент **useBSPAccessUuid** *localSubscription* имеет значение **FALSE**, компонент **bspUuid** *outgoingCallbackParams* должен быть установлен из компонента **bspProductUuid** *eventInfo*. В противном случае компонент должен быть установлен из компонента **bspAccessUuid** поля таблицы **VisibleBSPRegistrations** (см. 18.3), в котором компоненты **bspProductUuid** и **hostingEndpointIRI** поля имеют значения, аналогичные значениям компонентов *eventInfo* с теми же именами и

3) оставшиеся компоненты должны быть установлены из компонентов *eventInfo* с такими же именами; также следует выполнить вызов к функции обратного вызова **BioAPI_EVENT_HANDLER** локального приложения, в котором адрес обратного вызова и параметры вызова функции должны быть установлены путем преобразования из *outgoingCallbackParams* согласно 18.1.8. Проигнорировать значение, возвращенное каждым из этих вызовов, но ожидать перед совершением следующего вызова возвращения каждого предыдущего вызова.

29.3 Если компонент **hostingEndpointIRI** *eventInfo* содержит ИИР локальной конечной точки, для каждого поля (*masterEndpoint*) таблицы **MasterEndpoints** (см. 18.1) структура должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **UnitEventNotificationDisablers** (см. 18.7) на наличие поля, в котором:

1) компонент **referrerEndpointIRI** имеет значение, аналогичное значению компонента **masterEndpointIRI** *masterEndpoint*;

2) компонент **bspProductUuid** имеет значение, аналогичное значению компонента **bspProductUuid** *eventInfo* и

3) компонент **unitEventTypes** имеет значение, означающее, что тип операции идентификации *eventInfo* отключен;

b) в случае, если существует соответствующее поле, не выполнять следующие действия (продолжить со следующего поля таблицы **MasterEndpoints**);

c) проверить таблицу **RunningBSPRemoteReferences** (см. 18.6) на наличие поля, в котором:

1) компонент **referrerEndpointIRI** имеет значение, аналогичное значению компонента **masterEndpointIRI** *masterEndpoint*;

2) компонент **bspProductUuid** имеет значение, аналогичное значению компонента **bspProductUuid** *eventInfo* и

3) компонент **unitEventSubscription** имеет значение **TRUE**;

d) в случае, если соответствующее поле отсутствует, не выполнять следующие действия (продолжить с следующим полем таблицы **MasterEndpoints**);

e) создать временное абстрактное значение (*outgoingNotificationParams*) типа **UnitEvent-NotificationParams** (см. 18.1.3), в котором все компоненты должны быть установлены из компонентов *eventInfo* с такими же именами;

f) создать и отправить сообщение уведомления ПМО БиоАПИ **unitEvent** (см. 13.4 и 18.1.3) с ИИР главной конечной точки, установленным из компонента **masterEndpointIRI** *masterEndpoint*, и значением параметра установленным на *outgoingNotificationParams*.

Примечание – Не более одного сообщения должно быть отправлено в каждую главную конечную точку даже в том случае, если существует множество совпадающих полей в таблице **RunningBSPRemoteReferences** с одинаковыми главными конечными точками.

30 Предоставление подписчику информации об операции выбора ГИП

30.1 Данный раздел применяют в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для предоставления информации об операции выбора ГИП, основанной на значении (*eventInfo*) типа **GUISelectEventInfo** подписчику, которым может быть:

a) обработчик операции выбора ГИП локального приложения или

b) главная конечная точка,

а также для определения значения параметра уведомления (*incomingAcknowledgmentParams*) и возвращаемого значения уведомления, являющегося результатом уведомления.

30.2 Если при применении следующих подразделов, структура установит, что подписчиков нет, она должна создать:

a) временное абстрактное значение *incomingAcknowledgementParams* типа **GUISelectEvent- AcknowledgementParams** (см. 18.2.3), в котором компонент **selectedInstances** должен быть установлен из компонента **selectableInstances** *eventInfo* и компонент ответа установлен на значение по умолчанию и

b) временное абстрактное значение *incomingReturnValue* типа **BioAPI-RETURN** (см. 16.52), которое установлено на 0.

30.3 Если компонент **subscriberEndpointIRI** *eventInfo* содержит ИИР локальной конечной точки, структура должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **GUIEventLocalSubscriptions** (см. 18.10) на наличие поля (*localSubscription*), в котором:

1) компоненты **hostingEndpointIRI** и **bspProductUuid** имеют значения, аналогичные значениям компонентов *eventInfo* с такими же именами;

2) необязательный компонент **guiEventSubscriptionUuid** присутствует и имеет значение, аналогичное значению необязательного компонента **guiEventSubscriptionUuid** *eventInfo*;

3) в случае, если необязательный компонент **originalBSPHandle** присутствует, необязательный компонент **originalBSPHandle** *eventInfo* также должен присутствовать, и оба этих компонента должны иметь одинаковые значения и

4) компонент **guiSelectEventHandlerAddress** имеет значение, отличающееся от 0;

b) Если соответствующее поле отсутствует, удостовериться, что подписчиков нет (см. 30.2) и не выполнять следующие действия.

Примечание – В этом случае может быть не более одного соответствующего поля;

c) создать временное абстрактное значение (*outgoingCallbackParams*) типа **GUISelectEventHandlerCallbackParams** (см. 18.2.4), в котором:

1) компоненты **guiSelectEventHandlerAddress** и **guiSelectEventHandlerContext** должны быть установлены из компонентов *localSubscription* с теми же именами;

2) в случае, если компонент **useBSPAccessUuid** *localSubscription* имеет значение **FALSE**, компонент **bspUuid** *outgoingCallbackParams* должен быть установлен из компонента **bspProductUuid** *eventInfo*; в противном случае, данный компонент должен быть установлен из компонента **bspAccessUuid** поля таблицы **VisibleBSPRegistrations** (см. 18.3), в котором компоненты **bspProductUuid** и **hostingEndpointIRI** поля имеют значения, аналогичные значениям компонентов *eventInfo* с такими же именами;

3) в случае, если необязательный компонент **originalBSPHandle** *eventInfo* отсутствует, необязательный компонент **bspHandle** *outgoingCallbackParams* также должен отсутствовать; в противном случае, данный компонент должен быть установлен из компонента **localBSPHandle** поля таблицы **AttachSessionLocalReferences** (см. 18.8), в котором компоненты **originalBSPHandle** и **hostingEndpointIRI** поле имеют значения, аналогичные значениям компонентов *eventInfo* с такими же именами и

4) оставшиеся компоненты должны быть установлены из компонентов *eventInfo* с такими же именами;

d) выполнить вызов функции обратного вызова **BioAPI_GUI_SELECT_EVENT_HANDLER** локального приложения, в котором адрес обратного вызова и параметры вызова функции должны быть установлены путем преобразования из *outgoingCallbackParams* согласно 18.2.10;

e) создать временное абстрактное значение (*incomingAcknowledgementParams*) типа **GUISelectEvent-AcknowledgementParams** (см. 18.2.3) путем преобразования из исходящих параметров вызова функции согласно 18.2.12;

f) создать временное абстрактное значение (*incomingReturnValue*) типа **BioAPI-RETURN** (см. 18.52) путем преобразования из возвращенного значения вызова функции согласно 18.1.5.

30.4 Если компонент **subscriberEndpointIRI** *eventInfo* содержит ИИР главной конечной точки, структура должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **GUIEventRemoteSubscriptions** (см. 18.11) на наличие поля, в котором:

1) компоненты **subscriberEndpointIRI** and **bspProductUuid** имеют значения, аналогичные значениям компонентов *eventInfo* с такими же именами;

2) необязательный компонент **guiEventSubscriptionUuid** должен присутствовать и иметь значение, аналогичное значению необязательного компонента **guiEventSubscriptionUuid** *eventInfo*;

3) необязательный компонент **originalBSPHandle** должен присутствовать и иметь значение, аналогичное значению необязательного компонента **originalBSPHandle** *eventInfo* из

4) компонент **guiSelectEventSubscribed** имеет значение **TRUE**;

b) в случае, если соответствующее поле отсутствует, удостовериться, что подписчиков нет (см. 30.2) и не выполнять следующие действия;

c) создать временное абстрактное значение (*outgoingNotificationParams*) типа **GUISelectEvent-NotificationParams** (см. 18.2.3), в котором все компоненты должны быть установлены из компонентов *eventInfo* с такими же именами;

d) создать и отправить сообщение уведомления ПМО БиоАПИ **guiSelectEvent** (см. 13.4 и 18.2.3) с ИИР главной конечной точки, установленным из компонента **subscriberEndpointIRI** *eventInfo*, и значением параметра, установленным на *outgoingNotificationParams*;

e) принять соответствующее сообщение подтверждения ПМО БиоАПИ **guiSelectEvent** (см. 13.7);

f) разрешить *incomingAcknowledgementParams* иметь значение параметра типа **GUISelectEventAcknowledgementParams** (см. 18.2.3) сообщения подтверждения ПМО БиоАПИ **guiSelectEvent**;

g) разрешить *incomingReturnValue* иметь возвращаемое значение типа **BioAPI-RETURN** (см. 16.52) сообщения подтверждения ПМО БиоАПИ.

31 Предоставление подписчику информации об операции состояния ГИП

31.1 Данный раздел применяют только в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для предоставления информации об операции состояния ГИП, основанной на значении (*eventInfo*) типа **GUIStateEventInfo** подписчику, которым может быть:

a) обработчик операции состояния ГИП локального приложения или

b) главная конечная точка,

а также для определения значения параметра уведомления (*incomingAcknowledgementParams*) и возвращаемого значения уведомления, являющегося результатом уведомления.

31.2 Если при применении следующих подразделов, структура примет решение о том, что подписчиков нет, она должна создать:

a) временное абстрактное значение *incomingAcknowledgementParams* типа **GUIStateEvent- AcknowledgementParams** (см. 18.3.3), в котором компонент **enrollSampleIndexToRecapture** установлен на 0 и компонент ответа установлен на значение по умолчанию и

b) временное абстрактное значение *incomingReturnValue* типа **BioAPI-RETURN** (см. 16.52), установленное на 0.

31.3 Если компонент **subscriberEndpointIRI** *eventInfo* содержит ИИР локальной конечной точки, структура должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **GUIEventLocalSubscriptions** (см. 18.10) на наличие поля (*localSubscription*), в котором:

1) компоненты **hostingEndpointIRI** и **bspProductUuid** имеют такие же значения, как и компоненты *eventInfo* с такими же именами;

2) необязательный компонент **guiEventSubscriptionUuid** должен присутствовать и иметь значение, аналогичное значению необязательного компонента **guiEventSubscriptionUuid** *eventInfo*;

3) в случае, если необязательный компонент **originalBSPHandle** присутствует, необязательный компонент **originalBSPHandle** *eventInfo* также должен присутствовать, и оба этих компонента должны иметь одинаковые значения и

4) компонент **guiStateEventHandlerAddress** имеет значение, отличающееся от 0;

b) в случае, если соответствующее поле отсутствует, удостовериться, что подписчиков нет (см. 30.2) и не выполнять следующие действия.

Примечание – В этом случае может быть не более одного соответствующего поля;

c) создать временное абстрактное значение (*outgoingCallbackParams*) типа **GUIStateEventHandlerCallbackParams** (см. 18.3.4), в котором:

1) компоненты **guiStateEventHandlerAddress** и **guiStateEventHandlerContext** должны быть установлены из компонентов *localSubscription* с такими же именами;

2) в случае, если компонент **useBSPAccessUuid** *localSubscription* имеет значение **FALSE**, компонент **bspUuid** *outgoingCallbackParams* должен быть установлен из компонента **bspProductUuid** *eventInfo*; в противном случае данный компонент должен быть установлен из компонента **bspAccessUuid** поля таблицы **VisibleBSPRegistrations** (см. 18.3), в котором компоненты **bspProductUuid** и **hostingEndpointIRI** поля имеют значения, аналогичные значениям компонентов *eventInfo* с теми же именами;

3) в случае, если необязательный компонент **originalBSPHandle** *eventInfo* отсутствует, необязательный компонент **bspHandle** *outgoingCallbackParams* также должен отсутствовать; в противном случае данный компонент должен быть установлен из компонента **localBSPHandle** поля таблицы **AttachSessionLocalReferences** (см. 18.8), в котором компоненты **originalBSPHandle** и

hostingEndpointIRI поля имеют значения, аналогичные значениям компонентов *eventInfo* с такими же именами и

4) оставшиеся компоненты должны быть установлены из компонентов *eventInfo* с такими же именами;

d) выполнить вызов функции обратного вызова **BioAPI_GUI_STATE_EVENT_HANDLER** локального приложения, в котором адрес обратного вызова и параметры вызова функции должны быть установлены путем преобразования из *outgoingCallbackParams* согласно 18.3.10;

e) создать временное абстрактное значение (*incomingAcknowledgementParams*) типа **GUIStateEvent-AcknowledgementParams** (см. 18.3.3) путем преобразования исходящих параметров вызова функции согласно 18.3.12;

f) создать временное абстрактное значение (*incomingReturnValue*) типа **BioAPI-RETURN** (см. 16.52) путем преобразования возвращенного значения вызова функции согласно 16.1.5.

31.4 Если компонент **subscriberEndpointIRI** *eventInfo* содержит ИИР главной конечной точки, структура должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **GUIEventRemoteSubscriptions** (см. 18.11) на наличие поля, в котором:

1) компоненты **subscriberEndpointIRI** и **bspProductUuid** должны иметь значения, аналогичные значениям компонентов *eventInfo* с теми же именами;

2) необязательный компонент **guiEventSubscriptionUuid** должен присутствовать и иметь значение, аналогичное значению необязательного компонента **guiEventSubscriptionUuid** *eventInfo*;

3) необязательный компонент **originalBSPHandle** должен присутствовать и иметь значение, аналогичное значению необязательного компонента **originalBSPHandle** *eventInfo*; и

4) компонент **guiStateEventSubscribed** должен иметь значение **TRUE**;

b) в случае, если соответствующее поле отсутствует, удостовериться, что подписчиков нет (см. 30.2) и не выполнять жеследующие действия;

c) создать временное абстрактное значение (*outgoingNotificationParams*) типа **GUIStateEvent-NotificationParams** (см. 18.3.3), в котором все компоненты должны быть установлены из компонентов *eventInfo* с теми же именами;

d) создать и отправить сообщение уведомления ПМО БиоАПИ **guiStateEvent** (см. 13.4 и 18.3.3) с ИИР главной конечной точки, установленным из компонента **subscriberEndpointIRI** *eventInfo*, и значением параметра, установленным на *outgoingNotificationParams*;

e) принять соответсвующее сообщение подтверждения ПМО БиоАПИ **guiStateEvent** (см. 13.7);

f) разрешить *incomingAcknowledgementParams* иметь значение параметра типа **GUIStateEvent- AcknowledgementParams** (см. 18.3.3) сообщения подтверждения ПМО БиоАПИ **guiStateEvent**;

g) разрешить *incomingReturnValue* иметь значение параметра типа **BioAPI-RETURN** (см. 16.52) сообщения подтверждения ПМО БиоАПИ.

32 Предоставление подписчику информации об операции прогресса ГИП

32.1 Данный раздел применяют в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для предоставления информации об операции прогресса ГИП, основанной на значении (*eventInfo*) типа **GUIProgressEventInfo** подписчику, которым может быть:

a) обработчик операции прогресса ГИП локального приложения или

b) главная конечная точка,

а также для определения значения параметра уведомления (*incomingAcknowledgementParams*) и возвращаемого значения уведомления, являющегося результатом уведомления.

32.2 Если при применении следующих подразделов, структура примет решение о том, что подписчиков нет, она должна создать:

a) временное абстрактное значение *incomingAcknowledgementParams* типа **GUIProgressEvent-AcknowledgementParams** (см. 18.4.3), в котором компонент ответа установлен на значение по умолчанию и

b) временное абстрактное значение *incomingReturnValue* типа **BioAPI-RETURN** (см. 16.52), установленное на 0.

32.3 Если компонент **subscriberEndpointIRI** *eventInfo* содержит ИИР локальной конечной точки, структура должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **GUIEventLocalSubscriptions** (см. 18.10) на наличие поля (*localSubscription*), в котором:

1) компоненты **hostingEndpointIRI** и **bspProductUuid** имеют значения, аналогичные значениям компонентов *eventInfo* с теми же именами;

2) необязательный компонент **guiEventSubscriptionUuid** должен присутствовать и иметь значение, аналогичное значению необязательного компонента **guiEventSubscriptionUuid** *eventInfo*;

3) в случае, если необязательный компонент **originalBSPHandle** присутствует, необязательный компонент **originalBSPHandle** *eventInfo* также должны присутствовать, и оба этих компонента должны иметь одинаковые значения и

4) компонент **guiProgressEventHandlerAddress** должен иметь значение, отличающееся от 0;

b) в случае, если соответствующее поле отсутствует, удостовериться, что подписчиков нет (см. 30.2) и не выполнять следующие действия.

Примечание – В этом случае может быть не более одного соответствующего поля;

c) создать временное абстрактное значение (*outgoingCallbackParams*) типа **GUIProgressEventHandlerCallbackParams** (см. 18.4.4), в котором:

1) компоненты **guiProgressEventHandlerAddress** и **guiProgressEventHandlerContext** должны быть установлены из компонентов *localSubscription* с такими же именами;

2) в случае, если компонент **useBSPAccessUuid** *localSubscription* имеет значение **FALSE**, компонент **bspUuid** *outgoingCallbackParams* устанавливается из компонента **bspProductUuid** *eventInfo*; в противном случае, он должен быть установлен из компонента **bspAccessUuid** поля таблицы **VisibleBSPRegistrations** (см. 18.3), в котором компоненты **bspProductUuid** и **hostingEndpointIRI** поля имеют значения, аналогичные значениям компонентов *eventInfo* с теми же именами;

3) в случае, если необязательный компонент **originalBSPHandle** *eventInfo* отсутствует, необязательный компонент **bspHandle** *outgoingCallbackParams* также должен отсутствовать; в противном случае данный компонент должен быть установлен из компонента **localBSPHandle** поля таблицы **AttachSessionLocalReferences** (см. 18.8), в котором компоненты **originalBSPHandle** и **hostingEndpointIRI** поля имеют значения, аналогичные значениям компонентов *eventInfo* с теми же именами и

4) оставшиеся компоненты должны быть установлены из компонентов *eventInfo* с теми же именами;

d) выполнить вызов функции обратного вызова **BioAPI_GUI_PROGRESS_EVENT_HANDLER** локального приложения, в котором адрес обратного вызова и параметры вызова

функции должны быть установлены путем преобразования из *outgoingCallbackParams* согласно 18.4.10;

e) создать временное абстрактное значение (*incomingAcknowledgementParams*) типа **GUIProgressEvent-AcknowledgementParams** (см. 18.4.3) путем преобразования исходящих параметров вызова функции согласно 18.4.12;

f) создать временное абстрактное значение (*incomingReturnValue*) типа **BioAPI-RETURN** (см. 16.52) путем преобразования возвращенного значения вызова функции согласно 16.1.5.

32.4 Если компонент **subscriberEndpointIRI** *eventInfo* содержит ИИР главной конечной точки, структура должна выполнить следующие действия в указанном порядке:

a) проверить таблицу **GUIEventRemoteSubscriptions** (см. 18.11) на наличие поля, в котором:

1) компоненты **subscriberEndpointIRI** и **bspProductUuid** имеют значения, аналогичные значениям компонентов *eventInfo* с теми же именами;

2) необязательный компонент **guiEventSubscriptionUuid** должен присутствовать и иметь значение, аналогичное значению необязательного компонента **guiEventSubscriptionUuid** *eventInfo*;

3) необязательный компонент **originalBSPHandle** должен присутствовать и иметь значение, аналогичное значению необязательного компонента **originalBSPHandle** *eventInfo* и

4) компонент **guiProgressEventSubscribed** должен иметь значение **TRUE**;

b) в случае, если соответствующее поле отсутствует, удостовериться, что подписчиков нет (см. 30.2) и не выполнять следующие действия;

c) создать временное абстрактное значение (*outgoingNotificationParams*) типа **GUIProgressEventNotificationParams** (см. 18.4.3), в котором все

компоненты должны быть установлены из компонентов *eventInfo* с теми же именами;

d) создать и отправить сообщение уведомления ПМО БиоАПИ **guiProgressEvent** (см. 13.4 и 18.4.3) с ИИР главной конечной точки, установленным из компонента **subscriberEndpointIRI** *eventInfo*, и значением параметра, установленным на *outgoingNotificationParams*;

e) принять соответсвующее сообщение подтверждения ПМО БиоАПИ **guiProgressEvent** (см. 13.7);

f) разрешить *incomingAcknowledgementParams* иметь значение параметра типа **GUIProgressEvent-AcknowledgementParams** (см. 18.4.3) сообщения подтверждения ПМО БиоАПИ **guiProgressEvent**;

g) разрешить *incomingReturnValue* иметь значение параметра типа **BioAPI-RETURN** (см. 16.52) сообщения подтверждения ПМО БиоАПИ.

33 Обработка непреобразуемых значений Си

33.1 Данный раздел применяют в том случае, если на него имеется прямая ссылка в других разделах настоящего стандарта для обработки не преобразуемых значений Си, которые встречаются при преобразованиях из типа Си в тип АСН.1.

33.2 Если во время обработки полученного от главной конечной точки сообщения запроса ПМО БиоАПИ обнаружено непреобразуемое значение Си, структура должна создать и отправить соответствующее сообщение ответа (см. 13.3) с возвращаемым значением, установленным в **BioAPIERR_UNCONVERTIBLE_VALUE**, не завершая оставшийся процесс обработки.

33.3 Если во время обработки входящего вызова от локального приложения обнаружено не преобразуемое значение Си, структура должна возвратить значение **BioAPIERR_UNCONVERTIBLE_VALUE** локальному приложению, не завершая оставшийся процесс обработки.

33.4 Если во время обработки полученного от второстепенной конечной точки сообщения уведомления ПМО БиоАПИ обнаружено не преобразуемое значение Си и имеется связанный тип сообщения уведомления ПМО БиоАПИ, структура должна создать и отправить соответствующее сообщение уведомления ПМО БиоАПИ (см. 13.5) с возвращаемым значением, установленным в **BioAPIERR_UNCONVERTIBLE_VALUE**, не завершая оставшийся процесс обработки.

33.5 Если во время обработки входящего обратного вызова от ПБУ обнаружено непреобразуемое значение Си, структура должна возвратить значение **BioAPIERR_UNCONVERTIBLE_VALUE** ПБУ, не завершая оставшийся процесс обработки.

Приложение А
Спецификация привязки TCP/IP
(обязательное)

А.1 Общие положения

Спецификация протокола ПМО БиоАПИ относится к уровням 5 – 7 (слой сессии; слой представления; слой приложения) ISO/IEC OSI-7-уровневой модели с целью обеспечения его использования во всех сетях, поддерживающих все слои до транспортного. Настоящее приложение распространяется на привязку ПМО БиоАПИ к интерфейсу транспортного слоя, которая обеспечивается TCP/IP .

А.2 Сообщение транспортного уровня

Сообщение транспортного уровня для привязки, на которую распространяется данное приложение, определяется в АСН.1 следующим образом:

```
BIP-TCPIP {joint-iso-itu-t bip(41) modules(0) bip-tcpip(1) version1(1)}
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
IMPORTS BIPMessage FROM BIP{joint-iso-itu-t bip(41) modules(0) bip(0) version1(1)};

TCPIPBIPMessage ::= SEQUENCE {
        magicNumber             OCTET STRING(SIZE(4))('3AC49E70'H),
        version                 INTEGER{version-1(1)}(0..255),
        content                 CHOICE {
                bIPMessage      OCTET STRING(CONTAINING BIPMessage
                                ENCODED BY basic-per-aligned),
                keepalive       NULL,
                requestLinkChannelOnSpecifiedPort INTEGER(0..65535),
                requestLinkChannel NULL
                }
        }
basic-per-aligned OBJECT IDENTIFIER ::=
        {joint-iso-itu-t asn1(1) packed-encoding(3) basic(0) aligned(0)}
END
```

Альтернативные варианты компонента **content** определены в таблице А.1.

Таблица А.1 – Типы сообщений транспортного уровня в TCP/IP привязке ПМО БиоАПИ

Альтернативный вариант	Разрешенный отправитель сообщения	Цель сообщения	Содержание поля содержания
bipMessage	Любая конечная точка канала связи, который поддерживается соединением TCP/IP	Переносит закодированные сообщения ПМО БиоАПИ	Сообщение ПМО БиоАПИ (абстрактное значение АСН.1), закодированное в выровненный ПСК (см. ITU-T Rec. X.691 | ИСО/МЭК 8825-2)
Keepalive	Любая конечная точка канала связи, который поддерживается соединением TCP/IP	Информирует удаленную конечную точку о том, что отправляющая конечная точка все еще присутствует и находится в активном состоянии	Пусте поле
requestLinkChannel OnSpecifiedPort	Главная конечная точка в канале связи запроса/ответа, который поддерживается данным соединением TCP/IP (до тех пор пока канал связи уведомление/ подтверждение не будет установлен как часть той же связи ПМО БиоАПИ)	Обращение к второстепенной конечной точке с запросом об установлении канала связи уведомления/ подтверждения путем открытия отдельного соединения TCP/IP для определенного порта главной конечной точки	Двухбайтное целое поле без знака, содержащее номер порта
requestLinkChannel	Главная конечная точка в канале связи запроса/ответа, который поддерживается данным соединением TCP/IP (до тех пор пока канал связи уведомление/ подтверждение не будет установлен как часть той же связи ПМО БиоАПИ)	Обращение к второстепенной конечной точке с запросом об установлении канала связи уведомления/ подтверждения путем использования этого же TCP/IP, который будет поддерживать оба канала связи	Пустое поле

А.3 Соединение TCP/IP между двумя конечными точками ПМО БиоАПИ

А.3.1 Соединение TCP/IP между двумя конечными точками ПМО БиоАПИ может поддерживать любой из следующих каналов связи:

a) запроса/ответа;

b) уведомления/подтверждения или

c) запроса/ответа и уведомления/подтверждения, которые являются частью одной и той же связи ПМО БиоАПИ.

А.3.2 Рекомендуемым номером порта для соединения TCP/IP , поддерживающего канал связи запроса/ответа, является 4376. Рекомендуемым номером порта для соединения TCP/IP, поддерживающего канал связи уведомления/подтверждения, является 4376.

А.3.3 Сообщение транспортного уровня, содержащее альтернативу присутствия-активности (см. таблицу А.1), может быть отправлено с любого конца соединения TCP/IP. Частота таких сообщений определяется реализацией.

А.4 Роль конечной точки

А.4.1 Если структура должна использоваться для действия в роли в возможном канале связи запрос/ответ по соединению TCP/IP, она должна «прислушиваться» к входящим запросам соединения на рекомендуемый порт TCP/IP (см. А.3) или номер порт, определенный другими способами.

А.4.2 Если структура должна использоваться для действия в роли главной в канале связи запроса/ответа по соединению TCP/IP с заданной конечной точкой ПМО БиоАПИ, она должна открыть соединение TCP/IP с такой конечной точкой ПМО БиоАПИ, выполняя действия, указанные в либо рекомендованный номер порта (см. А.3), либо номер порта, определенный другими способами.

А.4.3 Если структура предназначена для действия в роли главной в канале связи запроса/ответа с заданной конечной точкой ПМО БиоАПИ (по любому транспортному протоколу) и канал связи уведомления/подтверждения

еще не установлен в этой связи ПМО БиоАПИ, структура может выполнить одно из следующих действий:

a) установить канал связи уведомления/подтверждения со второстепенной конечной точкой, выполняя действия, указанные в другой транспортный протокол; или

b) в случае, если канал связи запроса/ответа использует другой транспортный протокол, «прислушиваться» к входящим запросам соединения от второстепенной конечной точкой на рекомендуемый порт TCP/IP (см. A.3); или порт, определенный другими способами;

c) в случае, если используется канал связи запроса/ответа по TCP/IP, «прислушиваться» к входящим запросам соединения от второстепенной конечной точкой на произвольный порт TCP/IP с таким же IP адресом и отправить сообщение транспортного уровня, которое содержит альтернативный вариант **requestLinkChannelOnSpecifiedPort** (см. таблицу A.1), на второстепенную конечную точку, которая определяет номер порта, или

d) в случае, если используется канал связи запрос/ответ по TCP/IP, отправить сообщение транспортного уровня, которое содержит альтернативу **requestLinkChannel** (см. таблицу A.1), на второстепенную конечную точку, или

e) не выполнять никаких действий; в этом случае никакого канала связи уведомления/подтверждения связи ПМО БиоАПИ не будет.

A.4.4 Если структура играет роль второстепенной в канале связи запроса/ответа с заданной конечной точкой ПМО БиоАПИ по любому транспортному протоколу и канал связи уведомления/подтверждения еще не установлен в этой связи ПМО БиоАПИ, структура может выполнить одно из следующих действий:

a) установить канал связи уведомления/подтверждения с главной конечной точкой, выполняя действия, указанные в другой транспортный протокол или

b) в случае, если канал связи запроса/ответа использует другой транспортный протокол, открыть соединение TCP/IP с главной конечной точкой, выполняя действия, указанные в либо рекомендованный номер порта TCP/IP (см. A.3), либо номер порта, определенный другими способами;

c) в случае, если канал связи запроса/ответа по TCP/IP и сообщение транспортного уровня, которое содержит альтернативу **requestLinkChannelOnSpecifiedPort** (см. таблицу A.1), получено от главной конечной точки через такое соединение TCP/IP, открыть новое соединение TCP/IP с главной конечной точкой, выполняя действия, указанные в номер порта, указанный в сообщении транспортного уровня, которое содержит альтернативный вариант **requestLinkChannelOnSpecifiedPort** (см. таблицу A.1), полученную от главной конечной точки;

d) в случае, если канал связи запроса/ответа по TCP/IP и сообщение транспортного уровня, которое содержит альтернативный вариант **requestLinkChannel** (см. таблицу A.1), получено от главной конечной точки через такое соединение TCP/IP, отметить, что такое соединение TCP/IP будет поддерживать канал связи уведомления/подтверждения так же, как и канал связи запроса/ответа;

e) не выполнять действий; в этом случае канала связи уведомления/подтверждения в связи ПМО БиоАПИ не будет.

A.4.5 Структура, играющая роль главной, отправляет не более одного сообщения транспортного уровня, содержащего альтернативный вариант **requestLinkChannelOnSpecifiedPort** или **requestLinkChannel** (см. таблицу A.1), через соединение TCP/IP.

A.4.6 Если структура, играющая роль второстепенной, уже получила сообщение транспортного уровня, содержащее альтернативный вариант **requestLinkChannelOnSpecifiedPort** или **requestLinkChannel** (см. таблицу A.1) через соединение TCP/IP, она должна игнорировать последующие сообщения любого типа, поступающие через это соединение TCP/IP.

A.4.7 Структура, играющая роль главной в канале связи запроса/ответа по соединению TCP/IP, не должна закрывать это соединение TCP/IP до получения второстепенной конечной точки либо:

a) сообщения уведомления ПМО БиоАПИ **masterDeletionEvent**;

b) сообщения ответа ПМО БиоАПИ **deleteMaster** через канал связи запроса/ответа или

c) сообщения уведомления ПМО БиоАПИ **masterDeletionEvent** через канал связи уведомления/подтверждения, который является частью той же связи ПМО БиоАПИ независимо от транспортного протокола, который используется в таком канале связи.

A.4.8 Структура, играющая роль второстепенной в канале связи запроса/ответа по соединению TCP/IP, не должна закрывать это соединение TCP/IP до отправления на главную конечную точку сообщение ПМО БиоАПИ согласно A.4.7, перечисления a) и b).

A.4.9 Структура, играющая роль главной в канале связи уведомления/подтверждения по соединению TCP/IP, не должна закрывать соединение TCP/IP до получения второстепенной конечной точки либо:

a) сообщения ответа ПМО БиоАПИ **deleteMaster** через канал связи запроса/ответа, который является частью той же связи ПМО БиоАПИ независимо от транспортного протокола, который используется в таком канале связи, либо

b) сообщения уведомления ПМО БиоАПИ **masterDeletionEvent** через канал связи уведомления/подтверждения.

А.4.10 Структура, играющая роль второстепенной в канале связи уведомления/подтверждения по соединению TCP/IP, не должна закрывать это соединение TCP/IP до отправления на главную конечную точку сообщения ПМО БиоАПИ согласно А.4.9, перечисления a) и b).

А.4.11 Когда структура, играющая роль второстепенной в канале связи запроса/ответа по соединению TCP/IP, обнаружит сбой или преждевременное закрытие соединения TCP/IP, она должна выполнять действия, как будто получила сообщение запроса ПМО БиоАПИ **deleteMaster** от главной конечной точки.

А.4.12 Когда структура, играющая роль главной в канале связи уведомления/подтверждения по соединению TCP/IP, обнаружит сбой или преждевременное закрытие соединения TCP/IP, она должна выполнять действия, как будто получила сообщение уведомления ПМО БиоАПИ **masterDeletionEvent** от второстепенной конечной точки.

А.5 Закрытие соединения в результате ошибки

А.5.1 Если структура получает сообщение транспортного уровня, которое содержит альтернативный вариант **bipMessage**, содержащий закодированное сообщение ПМО БиоАПИ, и раскодирование сообщения ПМО БиоАПИ приводит к ошибке или раскодированное сообщение ПМО БиоАПИ оказывается неполным или короче, длины поля содержимого, конечная точка ПМО БиоАПИ должна закрыть соединение TCP/IP.

А.5.2 Если структура получает бесформенное сообщение транспортного уровня, она должна закрыть соединение TCP/IP.

А.6 Транспортировка сообщений ПМО БиоАПИ

Каждое сообщение ПМО БиоАПИ, отправляемое через канал связи по TCP/IP, должно переноситься внутри сообщения транспортного уровня, содержащего альтернативный вариант **bipMessage** согласно таблице А.1, и

быть закодировано в выровненные ПСК (см. ITU-T Rec. X.691 | ИСО/МЭК 8825-2).

А.7 Использование ИИР

ИИР конечных точек могут иметь любую форму и не принужденны привязкой TCP/IP (искл. см. А.4.4). Преобразование данных между ИИР и IP адресами в настоящем стандарте не рассматривается.

Приложение В

Спецификация обнаружения и объявления в привязке TCP/IP

(обязательное)

В.1 Общие положения

Настоящее приложение распространяется на дополнительные условия для TCP/IP привязки, указанные в приложении А.

В.2 Механизмы PnP

Механизмы PnP используют в том случае, когда системы ПМО БиоАПИ соединены с Ethernet-/ИП-основанной, которая соответствует следующим требованиям.

Физический слой и слой связи данных (слои 1 и 2)

Адаптер Ethernet должен соответствовать следующим требованиям:

a) поддержка – 10 Мбит;

b) опциональная поддержка – 100 Мбит и/или 1 Гбит операций;

c) поддержка режима автосогласования.

Слой сети (слой 3):

Использование протоколов IPv4 или IPv6;

Использование адреса слоя сети или имени сервера доменных имен для идентификации конечной точки ПМО БиоАПИ.

Транспортный слой (слой 4):

Использование протокола, пригодного для TCP и UDP.

Слои сессии, представления и приложения (слои 5 – 7):

Протоколы соединения с конечными точками ПМО БиоАПИ образованы в этих слоях и являются:

a) адресами IP и именами сервисов (см. раздел В.3 для IPv4 и раздел В.5 для IPv6);

b) сообщения ПМО БиоАПИ: протокол состоит из обмена ПМО БиоАПИ, указанного в разделе 12.

Сообщения критического времени/управление временем простоя и ошибками:

Выбор определенного времени определяют связанными протоколами соответствующими слоями. Управление передачей и ошибками связи осуществляется в слоях 1 – 4.

B.3 Установка адреса и имен в ИП4

B.3.1 Конфигурацию данных конечной точки ПМО БиоАПИ устанавливают с помощью одного из следующих механизмов:

a) DHCP (см. IETF RFC 2131);

b) динамической конфигурации IPv4 адреса локальной связи (см. IETF RFC 3927);

c) статической конфигурации.

B.3.2 Данные конфигурации должны иметь следующие значения:

a) адрес IPv4 локальной конечной точки ПМО БиоАПИ;

b) маску подсети локального приложения;

c) межсетевой интерфейс адреса IPv4;

d) не менее одного IP адреса сервера DNS.

Примечание – Широковещательный адрес локальной сети, необходимый для *сервиса обнаружения протокола* (см. B.7), может быть сформирован из IP адреса и маски подсети и, таким образом, он требует установки.

B.3.3 При использовании сервера DHCP, конечная точка ПМО БиоАПИ получает все необходимые данные либо путем статических листов, либо с помощью динамической конфигурации. Имя DNS, конфигурируемое в конечной точке ПМО БиоАПИ, может быть отправлено из конечной точки ПМО БиоАПИ на сервер DHCP, который также предоставляет расширения DNS-UPDATE для кооперации с конфигурируемым сервером DNS (см. IETF RFC 2136).

В.3.4 Динамическая конфигурация IPv4 адреса локальной связи (IETF RFC 3927), который также называют Авто-IP, является ПОА-основанным механизмом (см. IETF RFC 826), который пытается найти уникальные адреса без помощи центрального сервиса, такого как DHCP. Тем не менее IPv4 адреса в диапазоне от 169.254.1.0 до 169.254.254.255 (см. IETF RFC 3927) будут выбраны с помощью генератора случайных цифр.

Примечание – IETF RFC 3927 также определяет выбор определенного времени адресных проб и действий в случае конфликтов адресов.

В.3.5 Администратор через интерфейс управления выполняет конфигурацию статических адресов. Интерфейс пользователя должен предоставить указанные данные.

В.4 Функция конфигурации сети в IPv4

Примечание – См. рисунок В.1.

В.4.1 Конфигурируют конечную точку ПМО БиоАПИ для использования при статической или динамической сетевой конфигурации. Используемая конечная точка является выбором реализации. Заводские значения должны быть указаны в руководстве операции как стандартные. При статической конфигурации администратор несет ответственность за правильность всех установок.

Примечание – Неправильные установки могут привести к тому, что конечная точка ПМО БиоАПИ не будет обнаружена через сеть.

В.4.2 Во время динамической конфигурации, конечная точка ПМО БиоАПИ сначала совершает попытку получения данных конфигурации через сервер DHCP. Если в течение определенного времени ни один из серверов не отвечает, конечная точка ПМО БиоАПИ делает вывод, что среди них подходящих серверов нет. После обмена, конечная точка ПМО БиоАПИ устанавливает конфигурацию, выполняя действия, указанные в Авто-IP механизм (см. IETF RFC 3927). Авто-IP механизм определяет это время от времени снова, в результате сервер DHCP должен быть найден.

Примечание – Обработка ошибки (особенно обработка адреса конфликтует с ошибками аппаратного обеспечения) выполняется в TCP/IP стеке либо реализациями DHCP, либо Авто-IP механизмами.

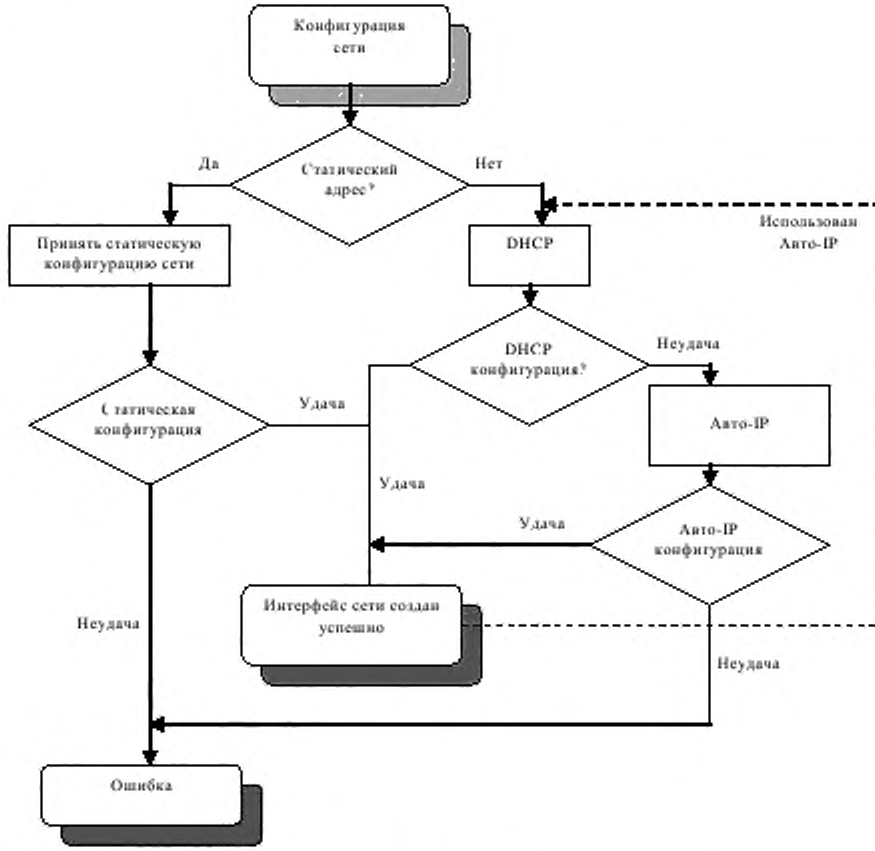


Рисунок В.1 – Функция конфигурации сети в IPv4

В.5 Установка адреса и имен в IPv6

В.5.1 Конечная точка ПМО БиоАПИ получает собственную конфигурацию данных с помощью одного из следующих механизмов:

a) механизма автоконфигурации stateful, DHCPv6 (см. IETF RFC 3315);

b) механизма автоконфигурации stateless (см. IETF RFC 2462);

c) статическойконфигурации.

В.5.2 Данные конфигурации должны содержать следующие значения:

a) глобальный однонаправленный IPv6 адрес конечной точки ПМО БиоАПИ;

b) длину префикса локальной сети;

c) не менее одного IPv6 адрес Сервера DNS.

Примечание – Адреса межсетевых интерфейсов приобретаются механизмом обнаружения маршрутизатора, которые являются частью ICMPv6 (см. IETF RFC 4443).

B.5.3 При использовании stateful автоконфигурации конечная точка ПМО БиоАПИ получает все необходимые данные от сервера DHCPv6 либо путем статических листов, либо с помощью динамической конфигурации. Имя DNS, конфигурируемое в конечной точке ПМО БиоАПИ, может быть отправлено из конечной точки ПМО БиоАПИ на сервер DHCPv6 (см. IETF RFC 3315), который также предоставляет расширения DNS-UPDATE для кооперации с конфигурируемым сервером DNS (см. IETF RFC 2136).

B.5.4 Механизм автоконфигурации stateless используют для приобретения глобального однонаправленного IPv6 адреса, когда в сети существует маршрутизатор IPv6. Сначала ПМО БиоАПИ конфигурирует IPv6 адрес локальной связи, полученный от Ethernet адреса аппаратного обеспечения интерфейса сети. Затем он отправляет сообщение с требованием (см. IETF RFC 3315) по определению маршрутизаторов, которые имеются в сети. Если маршрутизатор отвечает сообщением объявления, указывающим, что должен быть использован stateful, ПМО БиоАПИ попытается использовать DHCPv6 (см. IETF RFC 3315); Если ответ позволяет выполнить stateless автоконфигурацию, ПМО БиоАПИ создает глобальный однонаправленный IPv6 адрес, который объединяет IPv6 префикс, полученный от маршрутизатора, и собственный адрес Ethernet аппаратного обеспечения (см. IETF RFC 2462).

B.5.5 Администратор через интерфейс управления конечной точкой ПМО БиоАПИ выполняет конфигурацию статических адресов. Интерфейс пользователя должен предоставить вышеуказанные данные.

B.6 Функция конфигурации сети в IPv6

Примечание – См. рисунок B.2.

B.6.1 Конфигурируют конечную точку ПМО БиоАПИ для использования при статической или динамической сетевой конфигурации. Используемая точка, является выбором реализации. Заводские значения должны быть указаны в руководстве операции как стандартные. При статической конфигурации администратор несет ответственность за правильность всех установок.

Примечание – Неправильные установки могут привести к тому, что конечная точка ПМО БиоАПИ не будет обнаружена через сеть.

B.6.2 Во время динамической конфигурации, если была указана stateful автоконфигурация, конечная точка ПМО БиоАПИ совершает попытку получения данных конфигурации через сервер DHCPv6 (см. IETF RFC 3315). В противном случае, конечная точка ПМО БиоАПИ допускает объявления маршрутизатора. Если конечная точка получает объявление, указывающее stateful автоконфигурацию, она совершает попытку связаться с сервером DHCPv6; в противном случае, она создает собственный глобальный однонаправленный IPv6 адрес.

Примечание – Обработка ошибки (особенно обработка адреса конфликтует с ошибками аппаратного обеспечения) выполняют в TCP/IP стеке или реализациями протоколов автоконфигурации.

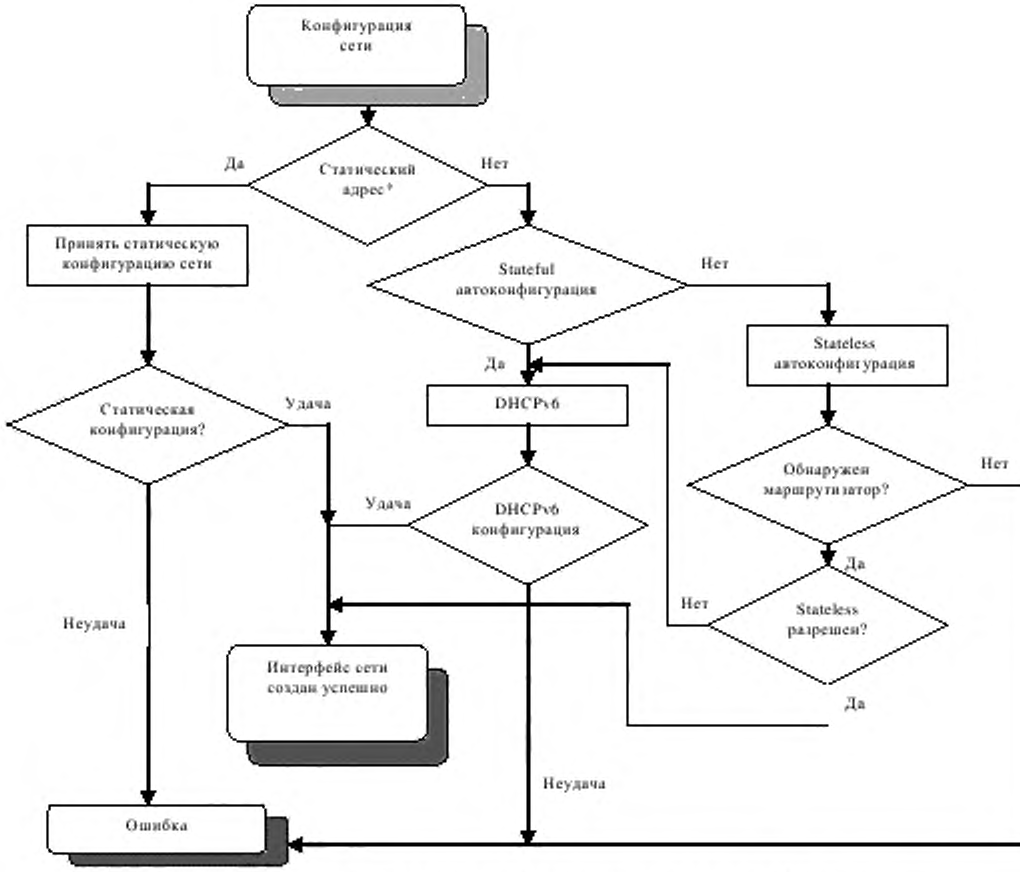


Рисунок В.2 – Функция конфигурации сети в IPv6

В.7 Обнаружение и объявление

Протоколы обнаружения конечных точек ПМО БиоАПИ находятся в слоях сессии, представления и приложения (слои 5 – 7). Они позволяют автоматически обнаруживать второстепенные конечные точки ПМО БиоАПИ через главные конечные точки ПМО БиоАПИ.

При использовании конечной точки ПМО БиоАПИ в сети необходима адресация трех зон:

а) адреса и установочного имени конечной точки ПМО БиоАПИ (конечная точка ПМО БиоАПИ содержит IP адрес и необязательно DNS имя). При этом другие параметры сети будут установлены;

b) динамического обнаружения структурой конечных точек ПМО БиоАПИ (услуга обнаружения): протокол, основанный на передаче или групповой передаче, позволяет приложениям получать действительные IP адреса и имена конечных точек. Данный этап выполняют дополнительно к коммуникации сообщений ПМО БиоАПИ с приложениями;

c) услуги обнаружения: дополнительно конечные точки ПМО БиоАПИ могут представить себя главным конечным точкам ПМО БиоАПИ.

Сообщения обнаружения ПМО БиоАПИ (услуга обнаружения/объявления) передается через незащищенные каналы. Когда канал коммуникации ПМО БиоАПИ будет защищен, один из протоколов безопасности, коммуникация через сообщения ПМО БиоАПИ начнется не ранее успешного запуска протокола безопасности. Таким образом можно избежать отказа сервиса.

При возможности ручной конфигурации конечных точек ПМО БиоАПИ, требование реализации услуги обнаружения и объявления не предъявляется (см. разделы B.8 и B.13). Для динамического обнаружения (см. раздел B.8) сервисный протокол обнаружения должен быть реализован, но услуга объявления является необязательной.

Второстепенная конечная точка ПМО БиоАПИ может показать главную конечную точку ПМО БиоАПИ для поддержки протоколов безопасности в поле «протоколы безопасности» в пакете услуги объявления.

B.8 Услуга обнаружения

B.8.1 Для обнаружения структурой ПМО БиоАПИ конечной точки ПМО БиоАПИ в сети используют UDP (см. IETF RFC 768), основанный на протоколе многоадресной передачи (или рассылки в IPv4). Протокол обнаружения не допускается использовать в локальной зоне сети. В настоящем приложении приведено описание версии 1.0 протокола обнаружения.

В.8.2 Услуга запроса выполняют по умолчанию через порт UDP 4376 (порт услуги запроса).

В.8.3 Главная конечная точка ПМО БиоАПИ отправляет запрос как многоадресную (или рассылку в IPv4) передачу.

В.8.4 Конечная точка ПМО БиоАПИ ожидает входящего запроса услуги на гнезде, сконфигурированном для принятия многоадресных сообщений.

Примечание – Прием таких запросов услуги возможно только в том случае, если режим передачи идентичен режиму конечной точки ПМО БиоАПИ.

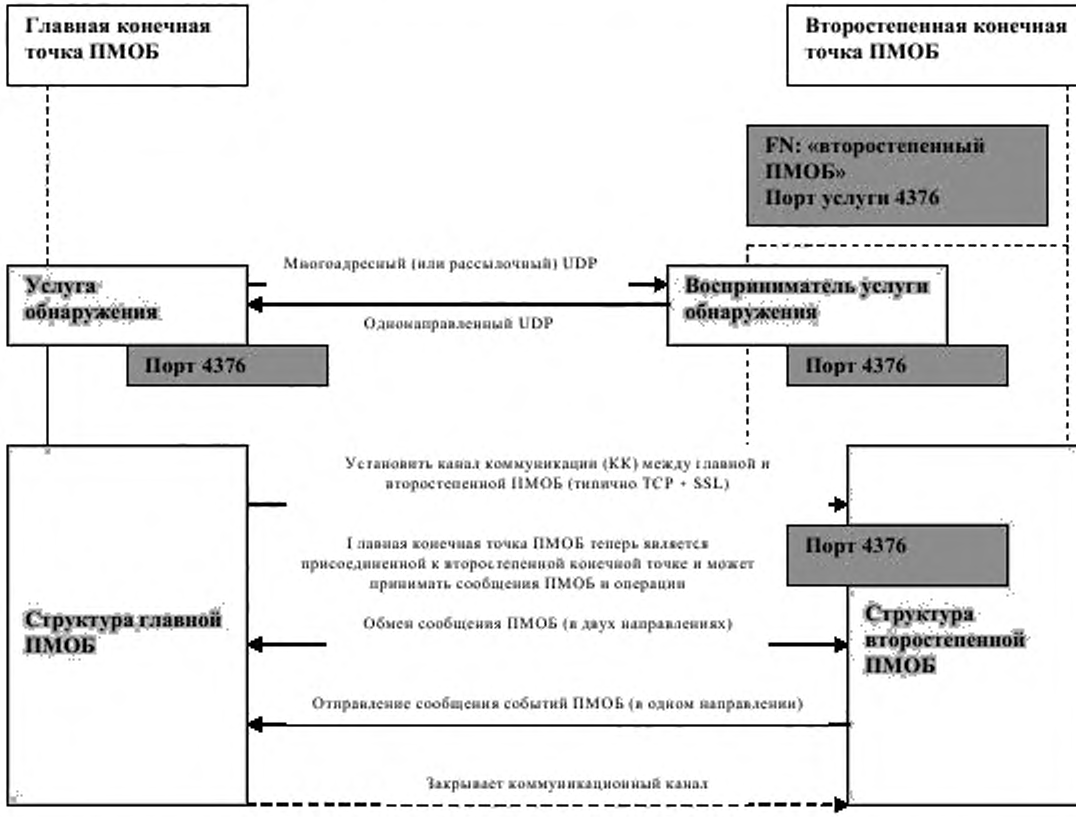


Рисунок В.3 – Принципиальная схема канала взаимодействия

В.9 Услуги запроса при рассылке (IPv4)

В.9.1 Рассылка может быть отправлена как направленные или локальные пакеты. Адрес цели рассылки контролирует ее вид:

a) рассылка в локальном приложении должна использовать адрес 255.255.255.255;

b) для направленной рассылки адрес может быть вычислен из IP адреса и маски подсети главной конечной точки ПМО БиоАПИ.

В.9.2 Для направленной рассылки в другую подсеть адрес рассылки должен быть конфигурируемым. Дополнительно сетевым администратором должно быть подтверждено, что рассылка будет отсортирована по маршруту в адрес цели и пакеты ответа могут также будут отсортированы по маршруту из адреса цели. Относящиеся к конфигурации сетевые компоненты (маршрутизатор, межсетевой экран и т. д.) в настоящем стандарте не рассматриваются.

В.9.3 При использовании рассылки, передатчик обычно не принимает собственные пакеты. Действия структуры ПМО БиоАПИ, как в случае главной, так и в случае второстепенной конечной точки, в этом случае требуют специальной обработки.

В.10 Услуги запроса при многоадресной передаче (IPv4 или IPv6)

При использовании многоадресной передачи группа адресов должна быть заменена соответствующим администрированием.

Сетевое администрирование должно подтвердить правильность передачи данных многоадресных сообщений ПМО БиоАПИ между подсетями.

В.11 Получение пакетов объявления сервиса

После передачи пакетов ответа сервиса, главная конечная ПМО БиоАПИ «прислушивается» к порту 4376 или порту, указанному в запросе сервиса, для пакета объявления сервиса от второстепенной конечной точки ПМО БиоАПИ. Время ожидания пакета объявления сервиса не ограничено.

Главная конечная точка ПМО БиоАПИ решает, как долго она будет ожидать этот тип сообщения.

Примечание – Для определения времени ожидания необходимо определить интенсивность передачи и использование сетей.

Каждая конечная точка ПМО БиоАПИ, которая получила пакет запроса сервиса, должна отправить на главную конечную точку ПМО БиоАПИ пакет объявления сервиса, с помощью UDP/IP: IP адрес цели и порт цели должны быть заданы в пакете запроса сервиса, допускается по умолчанию UDP – порт 4376.

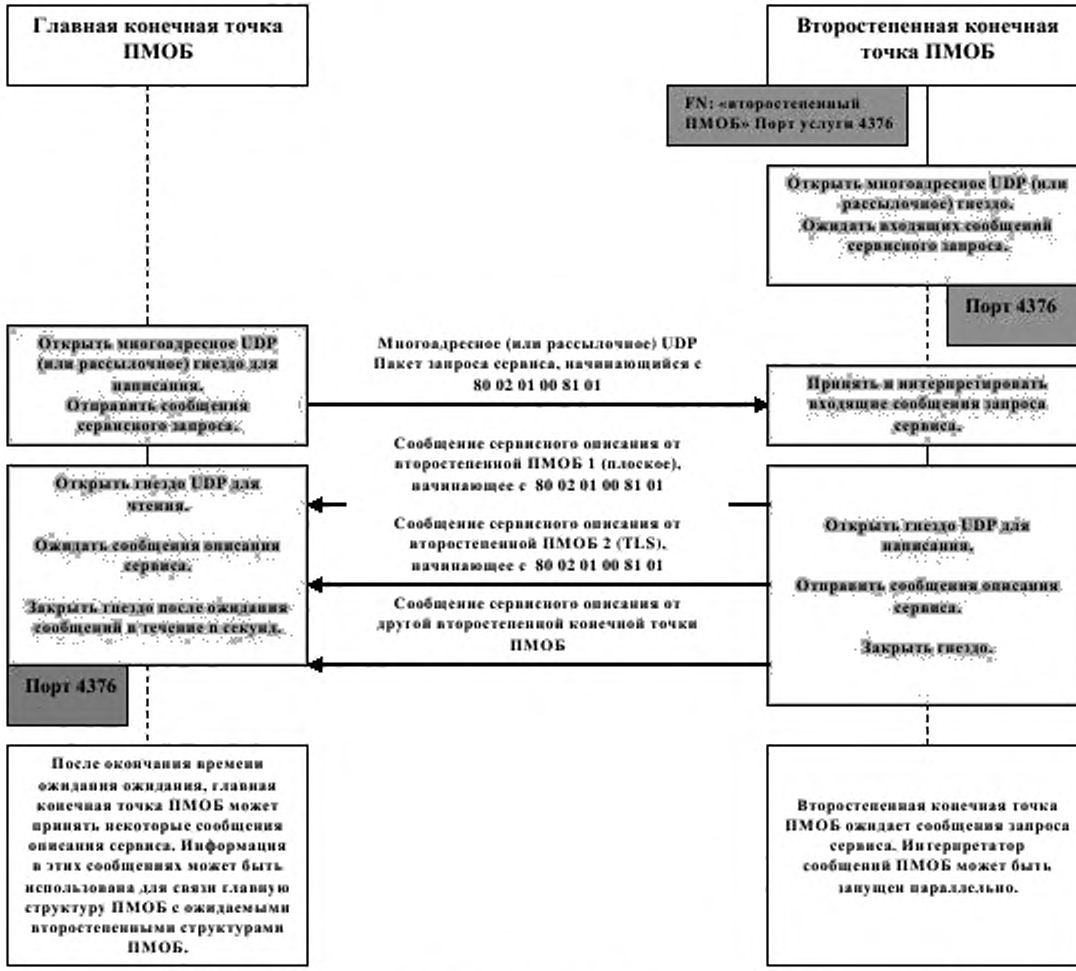


Рисунок В.4 – Услуга обнаружения рассылочная/многоадресная

В.12 Формат сообщений обнаружения и объявления

Сообщения обнаружения и объявления определены в следующем модуле ACH.1 и должны быть закодированы в выровненном ПСК (см. ITU-T Rec. X.691 | ИСО/МЭК 8825-2).

```
BIP-DISCOVERY {joint-iso-itu-t bip(41) modules(0) bip-discovery(2) version1(1)}
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

Discovery ::= SEQUENCE {
        protocolVersion             ProtocolVersion,
        masterEndpointAddress       IPAddress,
        masterEndPort                       Port DEFAULT 4376,
        ...
        }

Announcement ::= SEQUENCE {
        protocolVersion             ProtocolVersion,
        slaveEndpointIPAddress      IPAddress,
        slaveEndpointMACAddress     MACAddress,
        slaveEndpointName           IA5String(SIZE(1..32)),
        bipMessagePort Port         DEFAULT 4376,
        securityProtocols           SEQUENCE OF SecurityProtocol OPTIONAL,
        ...
        }

ProtocolVersion ::=     SEQUENCE {
        major           INTEGER(0..255),
        minor           INTEGER(0..255)
        }

PAddress ::=            CHOICE {
        ipv4            OCTET STRING(SIZE(4)),
        ipv6            OCTET STRING(SIZE(16))
        }

Port    ::=  INTEGER(0..65535)

MACAddress ::= OCTET STRING(SIZE(6))

SecurityProtocol        ::=SEQUENCE {
        id              SECURITY-PROTOCOL.&id({SecurityProtocols}),
        parameter       SECURITY-PROTOCOL.&Parameter({SecurityProtocols}{@id})
        }

SECURITY-PROTOCOL ::= CLASS {
```

```
    &id             OBJECT IDENTIFIER,
    &Parameter
    }

SecurityProtocols   SECURITY-PROTOCOL ::= {...}

END
```

В IPv4 сети, следует использовать альтернативный вариант **ipv4** для **IPAddress**, в IPv6 сети – **ipv6** для **IPAddress**.

Для компонента **major** должен быть один протокол, а для компонента **minor** – ни одного. Рекомендованным значением по умолчанию для порта является 4376. Порт может быть изменен путем использования интерфейса управления.

Примечание – Конечные точки ПМО БиоАПИ будут информированы о действительном порте сообщений ПМО БиоАПИ в пакете сервисного объявления.

Компонент **slaveEndpointMACAddress** является адресом интерфейса Ethernet конечных точек ПМО БиоАПИ (см. ИСО/МЭК ТО 8802-1).

Сообщения объявления могут содержать лист протоколов безопасности, поддерживаемых второстепенной конечной точкой. Отсутствие такого листа означает, что безопасность не поддерживается в передачах ПМО БиоАПИ. Каждый протокол безопасности определяется объектом ACH.1 как идентификатором значений и некоторых параметров.

B.13 Услуга объявления

Второстепенная конечная точка ПМО БиоАПИ может представить себя в сети путем отправления услуги объявления до получения пакета сервисного запроса. Для этого она отправляет пакет в порт 4376 (порт услуги объявления), выполняя действия, указанные в адрес рассылки в IPv4 или многоадресный адрес, определенный для услуги запроса IPv4 или IPv6.

Главные конечные точки, заинтересованные в пакетах сервисных объявлений, ожидают получения сообщения в порте 4376, а, в случае

многоадресного режима, должны «прислушиваться» к указанным многоадресным адресам.

Второстепенная конечная точка ПМО БиоАПИ не ожидает ответа от главной конечной точки ПМО БиоАПИ. Второстепенная конечная точка ПМО БиоАПИ должна отправлять пакеты сервисного объявления после любых изменений конфигурации, таких как:

a) изменение IP адреса;

b) изменение порта сообщений ПМО БиоАПИ;

c) изменение имени второстепенной конечной точки ПМО БиоАПИ.

Автономная передача сервисных объявлений позволяет главной конечной точке ПМО БиоАПИ определять изменения в конфигурации сети без уменьшения интервала передачи сервисных запросов рассылки или многоадресной передачи объявлений. Частая передача сервисных объявлений должна быть ограничена с целью предотвращения перегрузки сети.

B.14 Сброс и перезапуск

Изменение параметров настройки конфигурации и некоторые условия возникновения ошибок требуют завершения коммуникаций ПМО БиоАПИ и очистки внутренних состояний конечных точек ПМО БиоАПИ.

Когда новый параметр настройки активирован, для более быстрого завершения вышеуказанного второстепенная конечная точка ПМО БиоАПИ также может отправить сообщения сервисного объявления.

В худшем случае главная конечная точка ПМО БиоАПИ будет вынуждена отправлять сервисные сообщения обнаружения до тех пор, пока второстепенная конечная точка ПМО БиоАПИ не ответит на них.

Для завершения изменений параметров настройки конфигурации необходимо:

a) изменение IP адреса;

b) изменение порта сообщений ПМО БиоАПИ;

c) изменение параметров настройки безопасности для конечных точек ПМО БиоАПИ (включая соответствующие свидетельства).

Другие условия возникновения ошибки, которые требуют завершения соединений TCP/IP, указаны в разделе А.5.

После изменений обе стороны могут завершить соединение в соответствии с требованиями настоящего стандарта. После завершения соединения, новые параметры настройки будут активированы.

Примечание – Если было зафиксировано одно из вышеописанных состояний ошибки, второстепенная конечная точка ПМО БиоАПИ должна закрыть сетевое соединение и все внутренние состояния должны быть переведены в исходное состояние.

B.15 Определение времени обмена сообщениями по каналу связи

Структура второстепенной конечной точки ПМО БиоАПИ управляет определением времени.

Главные конечные точки должны принять в расчет тот факт, что для завершения некоторых операций может потребоваться значительное количество времени.

Главная конечная точка ПМО БиоАПИ может запросить статус обработки сообщения путем отправления сервисного сообщения запроса (см. 16.17).

Главная конечная точка ПМО БиоАПИ может завершить сообщения с помощью сообщения отмены (см. 16.57). Завершение сообщения может быть необходимым в случае, если:

a) время обработки превышает время, рассчитанное главной конечной точкой ПМО БиоАПИ;

b) возникла ошибка или условия аварийного прекращения передачи сообщения главной конечной точкой ПМО БиоАПИ.

B.16 Безопасность обмена сообщениями по каналу связи

Второстепенная конечная точка ПМО БиоАПИ ожидает, что согласования протокола начнется с самого приоритетного протокола. Если

главная конечная точка ПМО БиоАПИ не может установить соединение с помощью этого протокола, будет использован второй протокол по приоритетности.

Данная процедура протекает до тех пор, пока общий протокол не будет найден или не останется протоколов для согласования. Если ни один протокол не может быть согласован по TCP/IP, соединение будет закрыто второстепенной конечной точкой ПМО БиоАПИ.

Приложение С
Спецификация привязки SOAP/HTTP
(обязательное)

С.1 Общие условия

С.1.1 Настоящее приложение определяет использование ПМО БиоАПИ с протоколом SOAP, выполняя действия, указанные в HTTP или HTTPS в качестве переносчика.

С.1.2 Сообщение транспортного уровня для привязки, определенное в данном приложении, должно состоять из сообщения HTTP, которое переносит сообщение SOAP, закодированное в XML 1.0, в соответствии с привязкой HTTP, указанной в W3C SOAP, пункт 7.

С.1.3 Следует использовать версию SOAP 1.1 или 1.2.

С.1.4 Следует использовать версию HTTP 1.0 или 1.1. Допускается использовать HTTP по TLS (HTTPS).

С.1.5 Допускается использовать HTTP SOAP свойство оптимизации передачи, указанное в W3C SOAP MTOM, пункт 2.

Примечание – При использовании данного свойства любые бинарные данные (такие как ББД ПМО БиоАПИа), содержащиеся в сообщении ПМО БиоАПИ, могут быть включены в сообщение HTTP в качестве бинарного блока и не требует использования Base64.

С.1.6 Сообщение запроса ПМО БиоАПИ должно быть перенесено в сообщении запроса HTTP с HTTP с применением метода POST. Тело конверта SOAP внутри сообщения запроса HTTP должно содержать единичный экземпляр глобального элемента **request**, указанного в С.4.1. Заголовок HTTP **soapAction** (для SOAP 1.1) или параметр **action** типа медиа MIME (для SOAP 1.2) должны быть установлены в «**oid:/BIP/Reques”**".

С.1.7 Сообщение ответа ПМО БиоАПИ следует перенести в сообщении ответа HTTP, возвращенное второстепенной конечной точкой в ответ на сообщение запроса HTTP, содержащее соответствующее сообщение запроса ПМО БиоАПИ. Тело конверта SOAP в сообщении ответа HTTP должно

содержать единичный экземпляр глобального элемента **response**, указанного в С.4.2.

С.1.8 Сообщение уведомления ПМО БиоАПИ должно быть перенесено в сообщении запроса HTTP с HTTP методом POST. Тело конверта SOAP внутри сообщения запроса HTTP должно содержать единичный экземпляр глобального элемента **notification**, указанного в С.4.3. Заголовок HTTP **soapAction** HTTP (для SOAP 1.1) или параметр **action** типа медиа MIME (для SOAP 1.2) должны быть установлены в «**oid:/BIP/Notification**».

С.1.9 Сообщение подтверждения ПМО БиоАПИ должно быть перенесено в сообщении ответа HTTP, возвращенном второстепенной конечной точкой в ответ на сообщение запроса HTTP, содержащем соответствующее сообщение уведомления ПМО БиоАПИ. Тело конверта SOAP в сообщении ответа HTTP должно содержать единичный экземпляр глобального элемента **acknowledgement**, указанного в С.4.4. Для сообщения уведомления ПМО БиоАПИ, которое не подразумевает соответствующего сообщения подтверждения ПМО БиоАПИ, в главную конечную точку ПМО БиоАПИ должно быть отправлено сообщение ответа HTTP с пустым телом в ответ на запрос HTTP.

С.1.10 Семантика каждого определения типа XSD, описанного в данном приложении, определена путем интерпретации элемента такого типа с помощью EXTENDED-XER кодирования (см. ITU-T Rec. X.693/Amd.1 | ИСО/МЭК 8825-4/Amd.1) абстрактного значения типа ACH.1, который описан в соответствующих пунктах, на которые ссылаются, раздела 16 или 17. Условия пунктов, на которые ссылаются, косвенно относятся к созданию и обработке элементов такого типа XSD через данное семантическое преобразование данных.

С.2 Требования безопасности при использовании привязки SOAP/HTTP (учебный)

Существуют три разных способа применения методов безопасности в сообщениях ПМО БиоАПИ при использовании данной привязки:

a) может быть достигнуто обеспечение двухточечной (провайдерского уровня) безопасности путем использования HTTP по TLS (HTTPS) вместо регулярного HTTP по TCP/IP;

b) применение стандартного шифрования и методов целостности к целому сообщению XML, одному или нескольким его частям (таким как ЗБИ или ББД в ЗБИ, содержащегося в сообщении) путем XML шифрования или XML стандартов подписи, получения: W3C XMLENC и W3C XMLDSIG; или

c) использование стандартных условий безопасности БиоАПИ для шифрования ББД или подписания ЗБИ, содержащегося в сообщении ПМО БиоАПИ.

С.3 Заголовок схемы

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:bip="oid:/BIP"
targetNamespace="oid:/BIP"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
```

С.4 Глобальные элементы

С.4.1 Глобальный элемент request и тип BIPRequest

```
<xs:element name="request" type="BIPRequest"/>

  <xs:complexType name="BIPRequest">
    <xs:sequence>
      <xs:element name="masterEndpointIRI" type="EndpointIRI"/>
      <xs:element name="slaveEndpointIRI" type="EndpointIRI"/>
      <xs:element name="linkNumber" type="xs:unsignedInt"/>
      <xs:element name="requestId" type="xs:unsignedInt"/>
```

```
<xs:choice>
  <xs:element name="addMaster"
    type="AddMaster-RequestParams"/>
  <xs:element name="deleteMaster"
    type="DeleteMaster-RequestParams"/>
  <xs:element name="bspLoad"
    type="BSPLoad-RequestParams"/>
  <xs:element name="bspUnload"
    type="BSPUnload-RequestParams"/>
  <xs:element name="queryUnits"
    type="QueryUnits-RequestParams"/>
  <xs:element name="queryBFPs"
    type="QueryBFPs-RequestParams"/>
  <xs:element name="bspAttach"
    type="BSPAttach-RequestParams"/>
  <xs:element name="bspDetach"
    type="BSPDetach-RequestParams"/>
  <xs:element name="enableUnitEvents"
    type="EnableUnitEvents-RequestParams"/>
  <xs:element name="enableEventNotifications"
    type="EnableEventNotifications-RequestParams"/>
  <xs:element name="controlUnit"
    type="ControlUnit-RequestParams"/>
  <xs:element name="control"
    type="Control-RequestParams"/>
  <xs:element name="freeBIRHandle"
    type="FreeBIRHandle-RequestParams"/>
  <xs:element name="getBIRFromHandle"
    type="GetBIRFromHandle-RequestParams"/>
  <xs:element name="getHeaderFromHandle"
    type="GetHeaderFromHandle-RequestParams"/>
  <xs:element name="subscribeToGUIEvents"
    type="SubscribeToGUIEvents-RequestParams"/>
  <xs:element name="unsubscribeFromGUIEvents"
    type="UnsubscribeFromGUIEvents-RequestParams"/>
  <xs:element name="redirectGUIEvents"
    type="RedirectGUIEvents-RequestParams"/>
  <xs:element name="unredirectGUIEvents"
    type="UnredirectGUIEvents-RequestParams"/>
  <xs:element name="queryGUIEventSubscriptions"
    type="QueryGUIEventSubscriptions-RequestParams"/>
```

```
<xs:element name="notifyGUISelectEvent"
  type="NotifyGUISelectEvent-RequestParams"/>
<xs:element name="notifyGUIStateEvent"
  type="NotifyGUIStateEvent-RequestParams"/>
<xs:element name="notifyGUIProgressEvent"
  type="NotifyGUIProgressEvent-RequestParams"/>
<xs:element name="capture"
  type="Capture-RequestParams"/>
<xs:element name="createTemplate"
  type="CreateTemplate-RequestParams"/>
<xs:element name="process"
  type="Process-RequestParams"/>
<xs:element name="processWithAuxBIR"
  type="ProcessWithAuxBIR-RequestParams"/>
<xs:element name="verifyMatch"
  type="VerifyMatch-RequestParams"/>
<xs:element name="identifyMatch"
  type="IdentifyMatch-RequestParams"/>
<xs:element name="enroll"
  type="Enroll-RequestParams"/>
<xs:element name="verify"
  type="Verify-RequestParams"/>
<xs:element name="identify"
  type="Identify-RequestParams"/>
<xs:element name="import"
  type="Import-RequestParams"/>
<xs:element name="presetIdentifyPopulation"
  type="PresetIdentifyPopulation-RequestParams"/>
<xs:element name="transform"
  type="Transform-RequestParams"/>
<xs:element name="dbOpen"
  type="DbOpen-RequestParams"/>
<xs:element name="dbClose"
  type="DbClose-RequestParams"/>
<xs:element name="dbCreate"
  type="DbCreate-RequestParams"/>
<xs:element name="dbDelete"
  type="DbDelete-RequestParams"/>
<xs:element name="dbSetMarker"
  type="DbSetMarker-RequestParams"/>
<xs:element name="dbFreeMarker"
```

```
            type="DbFreeMarker-RequestParams"/>
          <xs:element name="dbStore"
            type="DbStoreBIR-RequestParams"/>
          <xs:element name="dbGetBIR"
            type="DbGetBIR-RequestParams"/>
          <xs:element name="dbGetNextBIR"
            type="DbGetNextBIR-RequestParams"/>
          <xs:element name="dbDeleteBIR"
            type="DbDeleteBIR-RequestParams"/>
          <xs:element name="calibrateSensor"
            type="CalibrateSensor-RequestParams"/>
          <xs:element name="setPowerMode"
            type="SetPowerMode-RequestParams"/>
          <xs:element name="setIndicatorStatus"
            type="SetIndicatorStatus-RequestParams"/>
          <xs:element name="getIndicatorStatus"
            type="GetIndicatorStatus-RequestParams"/>
          <xs:element name="cancel"
            type="Cancel-RequestParams"/>
          <xs:element name="registedBSP"
            type="RegisterBSP-RequestParams"/>
          <xs:element name="unregisterBSP"
            type="UnregisterBSP-RequestParams"/>
          <xs:element name="registerBFP"
            type="RegisterBFP-RequestParams"/>
          <xs:element name="unregisterBFP"
            type="UnregisterBFP-RequestParams"/>
        </xs:choice>
        <xs:any minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BIPRequest**, указанного в разделе 14.

С.4.2 Глобальный элемент response и тип BIPResponse

```
<xs:element name="response" type="BIPResponse"/>

<xs:complexType name="BIPResponse">
```

```
<xs:sequence>
  <xs:element name="slaveEndpointIRI" type="EndpointIRI"/>
  <xs:element name="masterEndpointIRI" type="EndpointIRI"/>
  <xs:element name="linkNumber" type="xs:unsignedInt"/>
  <xs:element name="requestId" type="xs:unsignedInt"/>
  <xs:choice>
    <xs:element name="addMaster"
      type="AddMaster-ResponseParams"/>
    <xs:element name="deleteMaster"
      type="DeleteMaster-ResponseParams"/>
    <xs:element name="bspLoad"
      type="BSPLoad-ResponseParams"/>
    <xs:element name="bspUnload"
      type="BSPUnload-ResponseParams"/>
    <xs:element name="queryUnits"
      type="QueryUnits-ResponseParams"/>
    <xs:element name="queryBFPs"
      type="QueryBFPs-ResponseParams"/>
    <xs:element name="bspAttach"
      type="BSPAttach-ResponseParams"/>
    <xs:element name="bspDetach"
      type="BSPDetach-ResponseParams"/>
    <xs:element name="enableUnitEvents"
      type="EnableUnitEvents-ResponseParams"/>
    <xs:element name="enableEventNotifications"
      type="EnableEventNotifications-ResponseParams"/>
    <xs:element name="controlUnit"
      type="ControlUnit-ResponseParams"/>
    <xs:element name="control"
      type="Control-ResponseParams"/>
    <xs:element name="freeBIRHandle"
      type="FreeBIRHandle-ResponseParams"/>
    <xs:element name="getBIRFromHandle"
      type="GetBIRFromHandle-ResponseParams"/>
    <xs:element name="getHeaderFromHandle"
      type="GetHeaderFromHandle-ResponseParams"/>
    <xs:element name="subscribeToGUIEvents"
      type="SubscribeToGUIEvents-ResponseParams"/>
    <xs:element name="unsubscribeFromGUIEvents"
      type="UnsubscribeFromGUIEvents-ResponseParams"/>
    <xs:element name="redirectGUIEvents"
```

```
    type="RedirectGUIEvents-ResponseParams"/>
  <xs:element name="unredirectGUIEvents"
    type="UnredirectGUIEvents-ResponseParams"/>
  <xs:element name="queryGUIEventSubscriptions"
    type="QueryGUIEventSubscriptions-ResponseParams"/>
  <xs:element name="notifyGUISelectEvent"
    type="NotifyGUISelectEvent-ResponseParams"/>
  <xs:element name="notifyGUIStateEvent"
    type="NotifyGUIStateEvent-ResponseParams"/>
  <xs:element name="notifyGUIProgressEvent"
    type="NotifyGUIProgressEvent-ResponseParams"/>
  <xs:element name="capture"
    type="Capture-ResponseParams"/>
  <xs:element name="createTemplate"
    type="CreateTemplate-ResponseParams"/>
  <xs:element name="process"
    type="Process-ResponseParams"/>
  <xs:element name="processWithAuxBIR"
    type="ProcessWithAuxBIR-ResponseParams"/>
  <xs:element name="verifyMatch"
    type="VerifyMatch-ResponseParams"/>
  <xs:element name="identifyMatch"
    type="IdentifyMatch-ResponseParams"/>
  <xs:element name="enroll"
    type="Enroll-ResponseParams"/>
  <xs:element name="verify"
    type="Verify-ResponseParams"/>
  <xs:element name="identify"
    type="Identify-ResponseParams"/>
  <xs:element name="import"
    type="Import-ResponseParams"/>
  <xs:element name="presetIdentifyPopulation"
    type="PresetIdentifyPopulation-ResponseParams"/>
  <xs:element name="transform"
    type="Transform-ResponseParams"/>
  <xs:element name="dbOpen"
    type="DbOpen-ResponseParams"/>
  <xs:element name="dbClose"
    type="DbClose-ResponseParams"/>
  <xs:element name="dbCreate"
    type="DbCreate-ResponseParams"/>
```

```
        <xs:element name="dbDelete"
          type="DbDelete-ResponseParams"/>
        <xs:element name="dbSetMarker"
          type="DbSetMarker-ResponseParams"/>
        <xs:element name="dbFreeMarker"
          type="DbFreeMarker-ResponseParams"/>
        <xs:element name="dbStore"
          type="DbStoreBIR-ResponseParams"/>
        <xs:element name="dbGetBIR"
          type="DbGetBIR-ResponseParams"/>
        <xs:element name="dbGetNextBIR"
          type="DbGetNextBIR-ResponseParams"/>
        <xs:element name="dbDeleteBIR"
          type="DbDeleteBIR-ResponseParams"/>
        <xs:element name="calibrateSensor"
          type="CalibrateSensor-ResponseParams"/>
        <xs:element name="setPowerMode"
          type="SetPowerMode-ResponseParams"/>
        <xs:element name="setIndicatorStatus"
          type="SetIndicatorStatus-ResponseParams"/>
        <xs:element name="getIndicatorStatus"
          type="GetIndicatorStatus-ResponseParams"/>
        <xs:element name="cancel"
          type="Cancel-ResponseParams"/>
        <xs:element name="registedBSP"
          type="RegisterBSP-ResponseParams"/>
        <xs:element name="unregisterBSP"
          type="UnregisterBSP-ResponseParams"/>
        <xs:element name="registerBFP"
          type="RegisterBFP-ResponseParams"/>
        <xs:element name="unregisterBFP"
          type="UnregisterBFP-ResponseParams"/>
      </xs:choice>
      <xs:element name="returnValue" type="BioAPI-RETURN"/>
      <xs:any minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BIPResponse**, указанного в разделе 14.

C.4.3 Глобальный элемент notification и тип BIPNotification

```
<xs:element name="notification" type="BIPNotification"/>

<xs:complexType name="BIPNotification">
  <xs:sequence>
    <xs:element name="slaveEndpointIRI" type="EndpointIRI"/>
    <xs:element name="masterEndpointIRI" type="EndpointIRI"/>
    <xs:element name="linkNumber" type="xs:unsignedInt"/>
    <xs:element name="notificationId" type="xs:unsignedInt"/>
    <xs:choice>
      <xs:element name="masterDeletionEvent"
        type="MasterDeletionEvent-NotificationParams"/>
      <xs:element name="unitEvent"
        type="UnitEvent-NotificationParams"/>
      <xs:element name="guiSelectEvent"
        type="GUISelectEvent-NotificationParams"/>
      <xs:element name="guiStateEvent"
        type="GUIStateEvent-NotificationParams"/>
      <xs:element name="guiProgressEvent"
        type="GUIProgressEvent-NotificationParams"/>
      <xs:element name="bspRegistrationEvent"
        type="BSPRegistrationEvent-NotificationParams"/>
      <xs:element name="bspUnregistrationEvent"
        type="BSPUnregistrationEvent-NotificationParams"/>
      <xs:element name="bfpRegistrationEvent"
        type="BFPRegistrationEvent-NotificationParams"/>
      <xs:element name="bfpUnregistrationEvent"
        type="BFPUnregistrationEvent-NotificationParams"/>
    </xs:choice>
    <xs:any minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BIPNotification**, указанного в разделе 14.

C.4.4 Глобальный элемент acknowledgement и тип BIPAcknowledgement

```
<xs:element name="acknowledgement" type="BIPAcknowledgement"/>

<xs:complexType name="BIPAcknowledgement">
  <xs:sequence>
    <xs:element name="masterEndpointIRI" type="EndpointIRI"/>
    <xs:element name="slaveEndpointIRI" type="EndpointIRI"/>
    <xs:element name="linkNumber" type="xs:unsignedInt"/>
    <xs:element name="notificationId" type="xs:unsignedInt"/>
    <xs:choice>
      <xs:element name="guiSelectEvent"
        type="GUISelectEvent-AcknowledgementParams"/>
      <xs:element name="guiStateEvent"
        type="GUIStateEvent-AcknowledgementParams"/>
      <xs:element name="guiProgressEvent"
        type="GUIProgressEvent-AcknowledgementParams"/>
    </xs:choice>
    <xs:element name="returnValue" type="BioAPI-RETURN"/>
    <xs:any minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BIPAcknowledgement**, указанного в разделе 14.

C.5 Типы

C.5.1 Тип EndpointIRI

```
<xs:simpleType name="EndpointIRI">
  <xs:restriction base="xs:token"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **EndpointIRI**, указанного в 15.3.

C.5.2 Тип BioAPI-BFP-LIST-ELEMENT

```
<xs:complexType name="BioAPI-BFP-LIST-ELEMENT">
  <xs:sequence>
    <xs:element name="category" type="BioAPI-CATEGORY"/>
    <xs:element name="bfpProductUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-BFP-LIST-ELEMENT**, указанного в 15.4.

C.5.3 Тип BioAPI-BFP-SCHEMA

```
<xs:complexType name="BioAPI-BFP-SCHEMA">
  <xs:sequence>
    <xs:element name="bfpProductUuid" type="BioAPI-UUID"/>
    <xs:element name="category" type="BioAPI-CATEGORY"/>
    <xs:element name="description" type="BioAPI-STRING"/>
    <xs:element name="path" type="xs:string"/>
    <xs:element name="specVersion" type="BioAPI-VERSION"/>
    <xs:element name="productVersion" type="BioAPI-STRING"/>
    <xs:element name="vendor" type="BioAPI-STRING"/>
    <xs:element name="supportedFormats">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="format" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"
              minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="factorsMask" type="BioAPI-BIR-BIOMETRIC-TYPE"/>
    <xs:element name="propertyUuid" type="BioAPI-UUID"/>
    <xs:element name="property" type="BioAPI-DATA"/>
    <xs:element name="hostingEndpointIRI" type="EndpointIRI"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-BFP-SCHEMA**, указанного в 15.5.

С.5.4 Тип BioAPI-BIR

```
<xs:complexType name="BioAPI-BIR">
  <xs:choice>
    <xs:element name="binaryBIR" type="xs:base64Binary"/>
    <xs:any namespace="##other" minOccurs="1" maxOccurs="1"/>
  </xs:choice>
  <xs:attribute name="patronFormatOwner" type="xs:unsignedShort" use="required"/>
  <xs:attribute name="patronFormatType" type="xs:unsignedShort" use="required"/>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-BIR**, указанного в 15.6 с последующей модификацией: абстрактное значение компонента **formattedBIR** (октетная строка) может либо быть закодировано в Base64 и «завернуто» в элемент **binaryBIR**, который является «потомком» элемента типа **BioAPI-BIR**, либо (если октетная строка является UTF-8 кодированием XML 1.0 элемента) может быть включено, как потомок элемента типа **BioAPI-BIR**.

Примечание – Рекомендуется использовать XML формата ведущей организации, указанный в ИСО/МЭК 19785-3.

С.5.5 Тип BioAPI-BIR-ARRAY-POPULATION

```
<xs:complexType name="BioAPI-BIR-ARRAY-POPULATION">
  <xs:sequence>
    <xs:element name="members">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="member" type="BioAPI-BIR"
                minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
```

```
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-BIR-ARRAY-POPULATION**, указанного в 15.7.

C.5.6 Тип BioAPI-BIR-BIOMETRIC-DATA-FORMAT

```
<xs:complexType name="BioAPI-BIR-BIOMETRIC-DATA-FORMAT">
  <xs:sequence>
    <xs:element name="formatOwner" type="xs:unsignedShort"/>
    <xs:element name="formatType" type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-BIR-BIOMETRIC-DATA-FORMAT**, указанного в 15.8.

C.5.7 Тип BioAPI-BIR-BIOMETRIC-PRODUCT-ID

```
<xs:complexType name="BioAPI-BIR-BIOMETRIC-PRODUCT-ID">
  <xs:sequence>
    <xs:element name="productOwner" type="xs:unsignedShort"/>
    <xs:element name="productType" type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-BIR-BIOMETRIC-PRODUCT-ID**, указанного в 15.9.

C.5.8 Тип BioAPI-BIR-BIOMETRIC-TYPE

```
<xs:simpleType name="BioAPI-BIR-BIOMETRIC-TYPE">
  <xs:list itemType="SingleBiometricType"/>
</xs:simpleType>
```

```
<xs:simpleType name="SingleBiometricType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="multipleBiometricTypes"/>
    <xs:enumeration value="face"/>
    <xs:enumeration value="voice"/>
    <xs:enumeration value="finger"/>
    <xs:enumeration value="iris"/>
    <xs:enumeration value="retina"/>
    <xs:enumeration value="handGeometry"/>
    <xs:enumeration value="signatureSign"/>
    <xs:enumeration value="keystroke"/>
    <xs:enumeration value="lipMovement"/>
    <xs:enumeration value="gait"/>
    <xs:enumeration value="vein"/>
    <xs:enumeration value="dna"/>
    <xs:enumeration value="ear"/>
    <xs:enumeration value="foot"/>
    <xs:enumeration value="scent"/>
    <xs:enumeration value="other"/>
    <xs:enumeration value="password"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-BIR-BIOMETRIC-TYPE**, указанного в 15.10.

C.5.9 Тип BioAPI-BIR-HANDLE

```
<xs:simpleType name="BioAPI-BIR-HANDLE">
  <xs:restriction base="xs:int"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-BIR-HANDLE**, указанного в 15.12.

С.5.10 Тип BioAPI-BIR-HEADER

```
<xs:complexType name="BioAPI-BIR-HEADER">
  <xs:choice>
    <xs:element name="binaryBIR" type="xs:base64Binary"/>
    <xs:any namespace="##other" minOccurs="1" maxOccurs="1"/>
  </xs:choice>
  <xs:attribute name="patronFormatOwner" type="xs:unsignedShort" use="required"/>
  <xs:attribute name="patronFormatType" type="xs:unsignedShort" use="required"/>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-BIR-HEADER**, указанного в 15.13 с последующей модификацией: абстрактное значение компонента **formattedBIR** (октетная строка) может быть либо закодировано в Base64 и «завернуто» в элемент **binaryBIR**, который является «потомком» элемента типа **BioAPI-BIR-HEADER**, или (если октетная строка является UTF-8 кодированием XML 1.0 элемента) может быть включено, как потомок элемента типа **BioAPI-BIR-HEADER**.

Примечание – Рекомендуется использовать XML формат ведущей организации, указанный в ИСО/МЭК 19785-3.

С.5.11 Тип BioAPI-BIR-PURPOSE

```
<xs:simpleType name="BioAPI-BIR-PURPOSE">
  <xs:restriction base="xs:token">
    <xs:enumeration value="verify"/>
    <xs:enumeration value="identify"/>
    <xs:enumeration value="enroll"/>
    <xs:enumeration value="enrollVerify"/>
    <xs:enumeration value="enrollIdentify"/>
    <xs:enumeration value="audit"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа ACH.1 **BioAPI-BIR-PURPOSE**, указанного в 15.14.

C.5.12 Тип BioAPI-BIR-SECURITY-BLOCK-FORMAT

```
<xs:complexType name="BioAPI-BIR-SECURITY-BLOCK-FORMAT">
  <xs:sequence>
    <xs:element name="formatOwner" type="xs:unsignedShort"/>
    <xs:element name="formatType" type="xs:unsignedShort"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-BIR-SECURITY-BLOCK-FORMAT**, указанного в 15.15.

C.5.13 Тип BioAPI-BIR-SUBTYPE

```
<xs:complexType name="BioAPI-BIR-SUBTYPE">
  <xs:sequence>
    <xs:element name="subtype" type="SingleSubtype"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="SingleSubtype">
    <xs:choice>
        <xs:element name="anySubtype" type="anySubtype"/>
        <xs:element name="veinOnlySubtype" type="veinOnySubtype"/>
     </xs:choice>
    </xs:complexType>

    <xs:simpleType name="anySubtype">
<xs:restriction base="xs:token">
    <xs:enumeration value="left"/>
    <xs:enumeration value="right"/>
    <xs:enumeration value="thumb"/>
    <xs:enumeration value="pointerFinger"/>
    <xs:enumeration value="middleFinger"/>
    <xs:enumeration value="ringFinger"/>
    <xs:enumeration value="littleFinger"/>
```

```
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="veinOnlySubtype">
 <xs:restriction base="xs:token">
     <xs:enumeration value="veinPalm"/>
     <xs:enumeration value="veinBackofhand"/>
     <xs:enumeration value="veinWrist"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-BIR-SUBTYPE**, указанного в 15.16.

С.5.14 Тип BioAPI-BIR-SUBTYPE-MASK

```
<xs:simpleType name="BioAPI-BIR-SUBTYPE-MASK">
  <xs:list itemType="SingleSubtype-mask"/>
</xs:simpleType>

<xs:simpleType name="SingleSubtype-mask">
 <xs:restriction base="xs:token">
     <xs:enumeration value="left"/>
     <xs:enumeration value="right"/>
     <xs:enumeration value="leftThumb"/>
     <xs:enumeration value="leftPointerFinger"/>
     <xs:enumeration value="leftMiddleFinger"/>
     <xs:enumeration value="leftRingFinger"/>
     <xs:enumeration value="leftLittleFinger"/>
     <xs:enumeration value="rigthtThumb"/>
     <xs:enumeration value="rightPointerFinger"/>
     <xs:enumeration value="rightMiddleFinger"/>
     <xs:enumeration value="rightRingFinger"/>
     <xs:enumeration value="rightLittleFinger"/>
     <xs:enumeration value="left-vein-palm"/>
     <xs:enumeration value="left-vein-backofhand"/>
     <xs:enumeration value="left-vein-wrist"/>
     <xs:enumeration value="rigth-vein-palm"/>
     <xs:enumeration value="right-vein-backofhand"/>
     <xs:enumeration value="right-vein-wrist"/>
```

```
</xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-BIR-SUBTYPE**, указанного в 15.17.

C.5.15 Тип BioAPI-BSP-SCHEMA

```
<xs:complexType name="BioAPI-BSP-SCHEMA">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="description" type="BioAPI-STRING"/>
    <xs:element name="path" type="xs:string" minOccurs="0"/>
    <xs:element name="specVersion" type="BioAPI-VERSION"/>
    <xs:element name="productVersion" type="BioAPI-STRING"/>
    <xs:element name="vendor" type="BioAPI-STRING"/>
    <xs:element name="supportedFormats">
     <xs:complexType>
       <xs:sequence>
           <xs:element name="format" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"
              minOccurs="0" maxOccurs="4294967295"/>
       </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="factorsMask" type="BioAPI-BIR-BIOMETRIC-TYPE"/>
    <xs:element name="operations" type="BioAPI-OPERATIONS-MASK"/>
    <xs:element name="options" type="BioAPI-OPTIONS-MASK"/>
    <xs:element name="payloadPolicy" type="BioAPI-FMR"/>
    <xs:element name="maxPayloadSize" type="xs:unsignedInt"/>
    <xs:element name="defaultVerifyTimeout" type="xs:int"/>
    <xs:element name="defaultIdentifyTimeout" type="xs:int"/>
    <xs:element name="defaultCaptureTimeout" type="xs:int"/>
    <xs:element name="defaultEnrollTimeout" type="xs:int"/>
    <xs:element name="defaultCalibrateTimeout" type="xs:int"/>
    <xs:element name="maxBSPDbSize" type="xs:unsignedInt"/>
    <xs:element name="maxIdentify" type="xs:unsignedInt"/>
    <xs:element name="hostingEndpointIRI" type="EndpointIRI"/>
    <xs:element name="bspAccessUuid" type="BioAPI-UUID" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-BSP-SCHEMA**, указанного в 15.19.

С.5.16 Тип BioAPI-CANDIDATE

```
<xs:complexType name="BioAPI-CANDIDATE">
  <xs:sequence>
    <xs:choice>
     <xs:element name="birInDatabase" type="BioAPI-UUID"/>
     <xs:element name="birInArray" type="xs:unsignedInt"/>
     <xs:element name="birInPresetArray" type="xs:unsignedInt"/>
     </xs:choice>
    <xs:element name="fmrAchieved" type="BioAPI-FMR"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-CANDIDATE**, указанного в 15.20.

С.5.17 Тип BioAPI-CATEGORY

```
<xs:simpleType name="BioAPI-CATEGORY">
  <xs:restriction base="xs:token">
    <xs:enumeration value="archive"/>
    <xs:enumeration value="comparisonAlgorithm"/>
    <xs:enumeration value="processingAlgorithm"/>
    <xs:enumeration value="sensor"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-CATEGORY**, указанного в 15.21.

C.5.18 Тип BioAPI-DATA

```
<xs:simpleType name="BioAPI-DATA">
  <xs:restriction base="xs:base64Binary"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 BioAPI-DATA, указанного в 15.22.

C.5.19 Тип BioAPI-DATE

```
<xs:complexType name="BioAPI-DATE">
  <xs:sequence>
    <xs:element name="year" type="xs:unsignedShort"/>
    <xs:element name="month" type="xs:unsignedByte"/>
  <xs:element name="day" type="xs:unsignedByte"/>
 </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-DATE**, указанного в 15.23.

C.5.20 Тип BioAPI-DB-ACCESS-TYPE

```
<xs:simpleType name="BioAPI-DB-ACCESS-TYPE">
  <xs:list itemType="SingleAccessType"/>
</xs:simpleType>

<xs:simpleType name="SingleAccessType">
  <xs:restriction base="xs:token">
    <xs:enumeration value="read"/>
    <xs:enumeration value="write"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа АСН.1 **BioAPI-DB-ACCESS-TYPE**, указанного в 15.24.

С.5.21 Тип BioAPI-DB-MARKER-HANDLE

```
<xs:simpleType name="BioAPI-DB-MARKER-HANDLE">
  <xs:restriction base="xs:unsignedInt"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-DB-MARKER-HANDLE**, указанного в 15.25.

С.5.22 Тип BioAPI-DB-HANDLE

```
<xs:simpleType name="BioAPI-DB-HANDLE">
  <xs:restriction base="xs:int"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-DB-HANDLE**, указанного в 15.26.

С.5.23 Тип BioAPI-DBBIR-ID

```
<xs:complexType name="BioAPI-DBBIR-ID">
  <xs:sequence>
    <xs:element name="dbHandler" type="BioAPI-DB-HANDLE"/>
    <xs:element name="keyValue" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-DBBIR-ID**, указанного в 15.27.

С.5.24 Тип BioAPI-DTG

```
<xs:complexType name="BioAPI-DTG">
  <xs:sequence>
```

```
    <xs:element name="date" type="BioAPI-DATE"/>
    <xs:element name="time" type="BioAPI-TIME"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-DTG**, указанного в 15.28.

C.5.25 Тип BioAPI-UNIT-EVENT-TYPE

```
<xs:simpleType name="BioAPI-UNIT-EVENT-TYPE">
  <xs:restriction base="xs:token">
    <xs:enumeration value="insert"/>
    <xs:enumeration value="remove"/>
    <xs:enumeration value="fault"/>
    <xs:enumeration value="sourcePresent"/>
    <xs:enumeration value="sourceRemoved"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-UNIT-EVENT-TYPE**, указанного в 15.30.

C.5.26 Тип BioAPI-UNIT-EVENT-TYPE-MASK

```
<xs:simpleType name="BioAPI-UNIT-EVENT-TYPE-MASK">
  <xs:list itemType="BioAPI-UNIT-EVENT-TYPE"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-UNIT-EVENT-TYPE-MASK**, указанного в 15.31.

C.5.27 Тип BioAPI-FMR

```
<xs:simpleType name="BioAPI-FMR">
```

```
    <xs:restriction base="xs:int"/>
  </xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-FMR**, указанного в 15.32.

С.5.28 Тип BioAPI-FRAMEWORK-SCHEMA

```
<xs:complexType name="BioAPI-FRAMEWORK-SCHEMA">
  <xs:sequence>
    <xs:element name="fwProductUuid" type="BioAPI-UUID"/>
    <xs:element name="description" type="BioAPI-STRING"/>
    <xs:element name="path" type="xs:string" minOccurs="0"/>
    <xs:element name="specVersion" type="BioAPI-VERSION"/>
    <xs:element name="productVersion" type="BioAPI-STRING"/>
    <xs:element name="vendor" type="BioAPI-STRING"/>
    <xs:element name="propertyUuid" type="BioAPI-UUID" minOccurs="0"/>
    <xs:element name="property" type="BioAPI-DATA" minOccurs="0"/>
    <xs:element name="hostingEndpointIRI" type="EndpointIRI"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-FRAMEWORK-SCHEMA**, указанного в 15.33.

С.5.29 Тип BioAPI-GUI-BITMAP

```
<xs:complexType name="BioAPI-GUI-BITMAP">
  <xs:sequence>
    <xs:element name="subtypeMask" type="BioAPI-BIR-SUBTYPE-MASK"/>
    <xs:element name="width" type="xs:unsignedInt"/>
    <xs:element name="height" type="xs:unsignedInt"/>
    <xs:element name="bitmap" type="BioAPI-DATA" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-GUI-BITMAP**, указанного в 15.34.

С.5.30 Тип BioAPI-GUI-BITMAP-ARRAY

```
<xs:complexType name="BioAPI-GUI-BITMAP-ARRAY">
  <xs:sequence>
    <xs:element name="members">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="member" type="BioAPI-GUI-BITMAP"
              minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-GUI-BITMAP-ARRAY**, указанного в 15.35.

С.5.31 Тип BioAPI-GUI-EVENT-SUBSCRIPTION

```
<xs:complexType name="BioAPI-GUI-EVENT-SUBSCRIPTION">
  <xs:sequence>
    <xs:element name="subscriberEndpointIRI" type="EndpointIRI"/>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"/>
    <xs:element name="guiEventsSubscribed">
      <xs:simpleType>
        <xs:list itemType="SingleGUIEvent"/>
      </xs:simpleType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="SingleGUIEvent">
  <xs:restriction base="xs:token">
    <xs:enumeration value="select"/>
```

```
      <xs:enumeration value="state"/>
      <xs:enumeration value="progress"/>
    </xs:restriction>
  </xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-GUI-EVENT-SUBSCRIPTION**, указанного в 15.36.

C.5.32 Тип BIOAPI-GUI-ENROLL-TYPE

```
<xs:simpleType name="BioAPI-GUI-ENROLL-TYPE">
  <xs:list>
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="testVerify"/>
        <xs:enumeration value="multipleCapture"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:list>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-ENROLL-TYPE**, указанного в 15.38.

C.5.33 Тип BioAPI-GUI-MOMENT

```
<xs:simpleType name="BioAPI-GUI-MOMENT">
  <xs:restriction base="xs:token">
    <xs:enumeration value="beforeStart"/>
    <xs:enumeration value="during"/>
    <xs:enumeration value="afterEnd"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-GUI-MOMENT**, указанного в 15.37.

С.5.34 Тип BioAPI-GUI-OPERATION

```
<xs:simpleType name="BioAPI-GUI-OPERATION">
  <xs:restriction base="xs:token">
    <xs:enumeration value="capture"/>
    <xs:enumeration value="process"/>
    <xs:enumeration value="createTemplate"/>
    <xs:enumeration value="verifyMatch"/>
    <xs:enumeration value="identifyMatch"/>
    <xs:enumeration value="verify"/>
    <xs:enumeration value="identify"/>
    <xs:enumeration value="enroll"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-GUI-OPERATION**, указанного в 15.39.

С.5.35 Тип BioAPI-GUI-RESPONSE

```
<xs:simpleType name="BioAPI-GUI-RESPONSE">
  <xs:restriction base="xs:token">
    <xs:enumeration value="default"/>
    <xs:enumeration value="startCycle"/>
    <xs:enumeration value="startSubop"/>
    <xs:enumeration value="continueSubop"/>
    <xs:enumeration value="nextSubop"/>
    <xs:enumeration value="opComplete"/>
    <xs:enumeration value="abortSubop"/>
    <xs:enumeration value="recapture"/>
    <xs:enumeration value="restartCycle"/>
    <xs:enumeration value="cancelOp"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-GUI-RESPONSE**, указанного в 15.40.

С.5.36 Тип BioAPI-GUI-SUBOPERATION

```
<xs:simpleType name="BioAPI-GUI-SUBOPERATION">
  <xs:restriction base="xs:token">
    <xs:enumeration value="capture"/>
    <xs:enumeration value="process"/>
    <xs:enumeration value="createTemplate"/>
    <xs:enumeration value="verifyMatch"/>
    <xs:enumeration value="identifyMatch"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-GUI-SUBOPERATION**, указанного в 15.41.

С.5.37 Тип BioAPI-HANDLE

```
<xs:simpleType name="BioAPI-HANDLE">
  <xs:restriction base="xs:unsignedInt"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-HANDLE**, указанного в 15.42.

С.5.38 Тип BioAPI-IDENTIFY-POPULATION

```
<xs:complexType name="BioAPI-IDENTIFY-POPULATION">
  <xs:choice>
    <xs:element name="birDataBase" type="BioAPI-DB-HANDLE"/>
    <xs:element name="birArray" type="BioAPI-BIR-ARRAY-POPULATION"/>
```

```
    <xs:element name="birPresetArray">
     <xs:complexType/>
    </xs:element>
  </xs:choice>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-IDENTIFY-POPULATION**, указанного в 15.43.

C.5.39 Тип BioAPI-INDICATOR-STATUS

```
<xs:simpleType name="BioAPI-INDICATOR-STATUS">
  <xs:restriction base="xs:token">
    <xs:enumeration value="accept"/>
    <xs:enumeration value="reject"/>
    <xs:enumeration value="ready"/>
    <xs:enumeration value="busy"/>
    <xs:enumeration value="failure"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-INDICATOR-STATUS**, указанного в 15.45.

C.5.40 Тип BioAPI-INPUT-BIR

```
<xs:complexType name="BioAPI-INPUT-BIR">
  <xs:choice>
    <xs:element name="birInDB" type="BioAPI-DBBIR-ID"/>
    <xs:element name="birInBSP" type="BioAPI-BIR-HANDLE"/>
    <xs:element name="bir" type="BioAPI-BIR"/>
  </xs:choice>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-INPUT-BIR**, указанного в 15.46.

C.5.41 Тип BioAPI-OPERATIONS-MASK

```
<xs:simpleType name="BioAPI-OPERATIONS-MASK">
  <xs:list itemType="SingleOperation"/>
</xs:simpleType>

<xs:simpleType name="SingleOperation">
  <xs:restriction base="xs:token">
    <xs:enumeration value="enableEvents"/>
    <xs:enumeration value="subscribeToGUIEvents"/>
    <xs:enumeration value="capture"/>
    <xs:enumeration value="createTemplate"/>
    <xs:enumeration value="process"/>
    <xs:enumeration value="processWithAuxBIR"/>
    <xs:enumeration value="verifyMatch"/>
    <xs:enumeration value="identifyMatch"/>
    <xs:enumeration value="enroll"/>
    <xs:enumeration value="verify"/>
    <xs:enumeration value="identify"/>
    <xs:enumeration value="import"/>
    <xs:enumeration value="presetIdentifyPopulation"/>
    <xs:enumeration value="databaseOperations"/>
    <xs:enumeration value="setPowerMode"/>
    <xs:enumeration value="setIndicatorStatus"/>
    <xs:enumeration value="getIndicatorStatus"/>
    <xs:enumeration value="calibrateSensor"/>
    <xs:enumeration value="utilities"/>
    <xs:enumeration value="queryUnits"/>
    <xs:enumeration value="queryBFPs"/>
    <xs:enumeration value="controlUnit"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа АСН.1 **BioAPI-OPERATIONS-MASK**, указанного в 15.48.

С.5.42 Тип BioAPI-OPTIONS-MASK

```
<xs:simpleType name="BioAPI-OPTIONS-MASK">
  <xs:list itemType="SingleOption"/>
</xs:simpleType>

<xs:simpleType name="SingleOption">
  <xs:restriction base="xs:token">
    <xs:enumeration value="raw"/>
    <xs:enumeration value="qualityRaw"/>
    <xs:enumeration value="qualityIntermediate"/>
    <xs:enumeration value="qualityProcessed"/>
    <xs:enumeration value="appGUI"/>
    <xs:enumeration value="guiProgressEvents"/>
    <xs:enumeration value="sourcePresent"/>
    <xs:enumeration value="payload"/>
    <xs:enumeration value="birSign"/>
    <xs:enumeration value="birEncrypt"/>
    <xs:enumeration value="templateUpdate"/>
    <xs:enumeration value="adaptation"/>
    <xs:enumeration value="binning"/>
    <xs:enumeration value="selfContainedDevice"/>
    <xs:enumeration value="moc"/>
    <xs:enumeration value="subtypeToCapture"/>
    <xs:enumeration value="sensorBFP"/>
    <xs:enumeration value="archiveBFP"/>
    <xs:enumeration value="comparisonBFP"/>
    <xs:enumeration value="processingBFP"/>
    <xs:enumeration value="coarseScores"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-OPTIONS-MASK**, указанного в 15.49.

C.5.43 Тип BioAPI-POWER-MODE

```
<xs:simpleType name="BioAPI-POWER-MODE">
  <xs:restriction base="xs:token">
    <xs:enumeration value="normal"/>
    <xs:enumeration value="detect"/>
    <xs:enumeration value="sleep"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-POWER-MODE**, указанного в 15.50.

C.5.44 Тип BioAPI-RETURN

```
<xs:simpleType name="BioAPI-RETURN">
  <xs:restriction base="xs:unsignedInt"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа, как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-RETURN**, указанного в 15.52.

C.5.45 Тип BioAPI-STRING

```
<xs:simpleType name="BioAPI-STRING">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-STRING**, указанного в 15.53.

C.5.46 Тип BioAPI-TIME

```
<xs:complexType name="BioAPI-TIME">
  <xs:sequence>
    <xs:element name="hour" type="xs:unsignedByte"/>
    <xs:element name="minute" type="xs:unsignedByte"/>
```

```
    <xs:element name="second" type="xs:unsignedByte"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-TIME**, указанного в 15.54.

С.5.47 Тип BioAPI-UNIT-ID

```
<xs:simpleType name="BioAPI-UNIT-ID">
  <xs:restriction base="xs:unsignedInt"/>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-UNIT-ID**, указанного в 15.55.

С.5.48 Тип BioAPI-UNIT-LIST-ELEMENT

```
<xs:complexType name="BioAPI-UNIT-LIST-ELEMENT">
  <xs:sequence>
    <xs:element name="category" type="BioAPI-CATEGORY"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-UNIT-LIST-ELEMENT**, указанного в 15.56.

С.5.49 Тип BioAPI-UNIT-SCHEMA

```
<xs:complexType name="BioAPI-UNIT-SCHEMA">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitManagerProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="category" type="BioAPI-CATEGORY"/>
```

```
      <xs:element name="unitProperties" type="BioAPI-UUID"/>
      <xs:element name="vendorInformation" type="BioAPI-STRING"/>
      <xs:element name="supportedUnitEvents" type="BioAPI-UNIT-EVENT-TYPE-MASK"/>
      <xs:element name="propertyUuid" type="BioAPI-UUID"/>
      <xs:element name="property" type="BioAPI-DATA"/>
      <xs:element name="hardwareVersion" type="BioAPI-STRING"/>
      <xs:element name="firmwareVersion" type="BioAPI-STRING"/>
      <xs:element name="softwareVersion" type="BioAPI-STRING"/>
      <xs:element name="hardwareSerialNumber" type="BioAPI-STRING"/>
      <xs:element name="authenticatedHardware" type="xs:boolean"/>
      <xs:element name="maxBSPDbSize" type="xs:unsignedInt"/>
      <xs:element name="maxIdentify" type="xs:unsignedInt"/>
    </xs:sequence>
  </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BioAPI-UNIT-SCHEMA**, указанного в 15.57.

C.5.50 Тип BioAPI-UUID

```
<xs:simpleType name="BioAPI-UUID">
  <xs:restriction base="xs:token">
    <xs:length value="36"/>
  </xs:restriction>
</xs:simpleType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения ACH.1 типа **BioAPI-UUID**, указанного в 15.58.

C.5.51 Тип BioAPI-VERSION

```
<xs:complexType name="BioAPI-VERSION">
  <xs:sequence>
    <xs:element name="major" type="xs:unsignedByte"/>
    <xs:element name="minor" type="xs:unsignedByte"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BioAPI-VERSION**, указанного в 15.59.

С.6 Параметры сообщений запроса ПМО БиоАПИ

С.6.1 Параметры сообщения запроса ПМО БиоАПИ addMaster

```
<xs:complexType name="AddMaster-RequestParams">
  <xs:sequence>
    <xs:element name="bipVersion" type="BioAPI-VERSION"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **AddMaster-RequestParams**, указанного в 16.4.

С.6.2 Параметры сообщения запроса ПМО БиоАПИ deleteMaster

```
<xs:complexType name="DeleteMaster-RequestParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DeleteMaster-RequestParams**, указанного в 16.5.

С.6.3 Параметры сообщения запроса ПМО БиоАПИ bspLoad

```
<xs:complexType name="BSPLoad-RequestParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitEventSubscription" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения АСН.1 типа **BSPLoad-RequestParams**, указанного в 16.9.

С.6.4 Параметры сообщения запроса ПМО БиоАПИ bspUnload

```
<xs:complexType name="BSPUnload-RequestParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitEventSubscription" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BSPUnload-RequestParams**, указанного в 16.10.

С.6.5 Параметры сообщения запроса ПМО БиоАПИ queryUnits

```
<xs:complexType name="QueryUnits-RequestParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **QueryUnits-RequestParams**, указанного в 16.11.

С.6.6 Параметры сообщения запроса ПМО БиоАПИ queryBFPs

```
<xs:complexType name="QueryBFPs-RequestParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа АСН.1 **QueryBFPs-RequestParams**, указанного в 16.12.

С.6.7 Параметры сообщения запроса ПМО БиоАПИ bspAttach

```
<xs:complexType name="BSPAttach-RequestParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="version" type="BioAPI-VERSION"/>
    <xs:element name="units">
      <xs:complexType>
        <xs:sequence>
            <xs:element name="unit" type="BioAPI-UNIT-LIST-ELEMENT"
              minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BSPAttach-RequestParams**, указанного в 16.13.

С.6.8 Параметры сообщения запроса ПМО БиоАПИ bspDetach

```
<xs:complexType name="BSPDetach-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **BSPDetach-RequestParams**, указанного в 16.14.

С.6.9 Параметры сообщения запроса ПМО БиоАПИ enableUnitEvents

```
<xs:complexType name="EnableUnitEvents-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="unitEvents" type="BioAPI-UNIT-EVENT-TYPE-MASK"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **EnableUnitEvents-RequestParams**, указанного в 16.15.

С.6.10 Параметры сообщения запроса ПМО БиоАПИ enableEventNotifications

```
<xs:complexType name="EnableEventNotifications-RequestParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitEventTypes" type="BioAPI-UNIT-EVENT-TYPE-MASK"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **EnableEventNotifications-RequestParams**, указанного в 16.16.

С.6.11 Параметры сообщения запроса ПМО БиоАПИ controlUnit

```
<xs:complexType name="ControlUnit-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="controlCode" type="xs:unsignedInt"/>
    <xs:element name="inputData" type="BioAPI-DATA"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа, как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **ControlUnit-RequestParams**, указанного в 16.17.

С.6.12 Параметры сообщения ПМО БиоАПИ controlRequest

```
<xs:complexType name="Control-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="controlCode" type="BioAPI-UUID"/>
    <xs:element name="inputData" type="BioAPI-DATA"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **Control-RequestParams**, указанного в 16.18.

С.6.13 Параметры сообщения запроса ПМО БиоАПИ freeBIRHandle

```
<xs:complexType name="FreeBIRHandle-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="birHandle" type="BioAPI-BIR-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **FreeBIRHandle-RequestParams**, указанного в 16.19.

С.6.14 Параметры сообщения запроса ПМО БиоАПИ getBIRFromHandle

```
<xs:complexType name="GetBIRFromHandle-RequestParams">
```

```
    <xs:sequence>
      <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
      <xs:element name="birHandle" type="BioAPI-BIR-HANDLE"/>
    </xs:sequence>
  </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GetBIRFromHandle-RequestParams**, указанного в 16.20.

С.6.15 Параметры сообщения запроса ПМО БиоАПИ getHeaderFromHandle

```
  <xs:complexType name="GetHeaderFromHandle-RequestParams">
    <xs:sequence>
      <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
      <xs:element name="birHandle" type="BioAPI-BIR-HANDLE"/>
    </xs:sequence>
  </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GetHeaderFromHandle-RequestParams**, указанного в 16.21.

С.6.16 Параметры сообщения запроса ПМО БиоАПИ subscribeToGUIEvents

```
  <xs:complexType name="SubscribeToGUIEvents-RequestParams">
    <xs:sequence>
      <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"
          minOccurs="0"/>
      <xs:element name="bspProductUuid" type="BioAPI-UUID"
          minOccurs="0"/>
      <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"
          minOccurs="0"/>
      <xs:element name="guiSelectEventSubscribed" type="xs:boolean"/>
      <xs:element name="guiStateEventSubscribed" type="xs:boolean"/>
      <xs:element name="guiProgressEventSubscribed" type="xs:boolean"/>
```

```
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **SubscribeToGUIEvents-RequestParams**, указанного в 16.22.

С.6.17 Параметры сообщения запроса ПМО БиоАПИ unsubscribeFromGUIEvents

```
<xs:complexType name="UnsubscribeFromGUIEvents-RequestParams">
  <xs:sequence>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"
        minOccurs="0"/>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"
        minOccurs="0"/>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"
        minOccurs="0"/>
    <xs:element name="guiSelectEventSubscribed" type="xs:boolean"/>
    <xs:element name="guiStateEventSubscribed" type="xs:boolean"/>
    <xs:element name="guiProgressEventSubscribed" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **UnsubscribeFromGUIEvents-RequestParams**, указанного в 16.23.

С.6.18 Параметры сообщения запроса ПМО БиоАПИ queryGUIEventSubscriptions

```
<xs:complexType name="QueryGUIEventSubscriptions-RequestParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **QueryGUIEventSubscriptions-RequestParams**, указанного в 16.24.

С.6.19 Параметры сообщения запроса ПМО БиоАПИ notifyGUISelectEvent

```
<xs:complexType name="NotifyGUISelectEvent-RequestParams">
  <xs:sequence>
    <xs:element name="subscriberEndpointIRI" type="EndpointIRI"/>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"/>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="enrollType" type="BioAPI-GUI-ENROLL-TYPE"/>
    <xs:element name="operation" type="BioAPI-GUI-OPERATION"/>
    <xs:element name="moment" type="BioAPI-GUI-MOMENT"/>
    <xs:element name="resultCode" type="BioAPI-RETURN"/>
    <xs:element name="maxNumEnrollSamples" type="xs:unsignedInt"/>
    <xs:element name="selectableInstances" type="BioAPI-BIR-SUBTYPE-MASK"/>
    <xs:element name="capturedInstances" type="BioAPI-BIR-SUBTYPE-MASK"/>
    <xs:element name="text" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **NotifyGUISelectEvent-RequestParams**, указанного в 16.25.

С.6.20 Параметры сообщения запроса ПМО БиоАПИ notifyGUIStateEvent

```
<xs:complexType name="NotifyGUIStateEvent-RequestParams">
  <xs:sequence>
    <xs:element name="subscriberEndpointIRI" type="EndpointIRI"/>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"/>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
```

```
    <xs:element name="operation" type="BioAPI-GUI-OPERATION"/>
    <xs:element name="suboperation" type="BioAPI-GUI-SUBOPERATION"/>
    <xs:element name="purpose" type="BioAPI-BIR-PURPOSE"/>
    <xs:element name="moment" type="BioAPI-GUI-MOMENT"/>
    <xs:element name="resultCode" type="BioAPI-RETURN"/>
    <xs:element name="enrollSampleIndex" type="xs:int"/>
    <xs:element name="bitmaps" type="BioAPI-GUI-BITMAP-ARRAY" minOccurs="0"/>
    <xs:element name="text" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **NotifyGUIStateEvent-RequestParams**, указанного в 16.26.

C.6.21 Параметры сообщения запроса ПМО БиоАПИ notifyGUIProgressEvent

```
<xs:complexType name="NotifyGUIProgressEvent-RequestParams">
  <xs:sequence>
    <xs:element name="subscriberEndpointIRI" type="EndpointIRI"/>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"/>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="operation" type="BioAPI-GUI-OPERATION"/>
    <xs:element name="suboperation" type="BioAPI-GUI-SUBOPERATION"/>
    <xs:element name="purpose" type="BioAPI-BIR-PURPOSE"/>
    <xs:element name="moment" type="BioAPI-GUI-MOMENT"/>
    <xs:element name="suboperationProgress" type="xs:unsignedByte"/>
    <xs:element name="bitmaps" type="BioAPI-GUI-BITMAP-ARRAY" minOccurs="0"/>
    <xs:element name="text" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа ACH.1 **NotifyGUIProgressEvent-RequestParams**, указанного в 16.27.

С.6.22 Параметры сообщений запроса ПМО БиоАПИ redirectGUIEvents

```
<xs:complexType name="RedirectGUIEvents-RequestParams">
  <xs:sequence>
    <xs:element name="subscriberEndpointIRI" type="EndpointIRI"/>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"/>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="guiSelectEventRedirected" type="xs:boolean"/>
    <xs:element name="guiStateEventRedirected" type="xs:boolean"/>
    <xs:element name="guiProgressEventRedirected" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **RedirectGUIEvents-RequestParams**, указанного в 16.28.

С.6.23 Параметры сообщений запроса ПМО БиоАПИ unredirectGUIEvents

```
<xs:complexType name="UnredirectGUIEvents-RequestParams">
  <xs:sequence>
    <xs:element name="subscriberEndpointIRI" type="EndpointIRI"/>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"/>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="guiSelectEventRedirected" type="xs:boolean"/>
    <xs:element name="guiStateEventRedirected" type="xs:boolean"/>
    <xs:element name="guiProgressEventRedirected" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа АСН.1 **UnredirectGUIEvents-RequestParams**, указанного в 16.29.

С.6.24 Параметры сообщений запроса ПМО БиоАПИ capture

```
<xs:complexType name="Capture-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="purpose" type="BioAPI-BIR-PURPOSE"/>
    <xs:element name="subtype" type="BioAPI-BIR-SUBTYPE"/>
    <xs:element name="outputFormat" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"
        minOccurs="0"/>
    <xs:element name="timeout" type="xs:int"/>
    <xs:element name="no-auditData" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **Capture-RequestParams**, указанного в 16.30.

С.6.25 Параметры сообщений запроса ПМО БиоАПИ createTemplate

```
<xs:complexType name="CreateTemplate-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="capturedBIR" type="BioAPI-INPUT-BIR"/>
    <xs:element name="referenceTemplate" type="BioAPI-INPUT-BIR"
        minOccurs="0"/>
    <xs:element name="outputFormat" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"
        minOccurs="0"/>
    <xs:element name="payload" type="BioAPI-DATA" minOccurs="0"/>
    <xs:element name="no-templateUuid" type="xs:boolean" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа АСН.1 **CreateTemplate-RequestParams**, указанного в 16.31.

С.6.26 Параметры сообщения запроса ПМО БиоАПИ process

```
<xs:complexType name="Process-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="capturedBIR" type="BioAPI-INPUT-BIR"/>
    <xs:element name="outputFormat" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"
        minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **Process-RequestParams**, указанного в 16.32.

С.6.27 Параметры сообщения запроса ПМО БиоАПИ processWithAuxBIR

```
<xs:complexType name="ProcessWithAuxBIR-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="capturedBIR" type="BioAPI-INPUT-BIR"/>
    <xs:element name="auxiliaryData" type="BioAPI-INPUT-BIR"/>
    <xs:element name="outputFormat" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"
        minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **ProcessWithAuxBIR-RequestParams**, указанного в 16.33.

С.6.28 Параметры сообщения запроса ПМО БиоАПИ verifyMatch

```
<xs:complexType name="VerifyMatch-RequestParams">
```

```
    <xs:sequence>
      <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
      <xs:element name="maxFMRRequested" type="BioAPI-FMR"/>
      <xs:element name="processedBIR" type="BioAPI-INPUT-BIR"/>
      <xs:element name="referenceTemplate" type="BioAPI-INPUT-BIR"/>
      <xs:element name="no-adaptedBIR" type="xs:boolean"/>
      <xs:element name="no-fmrAchieved" type="xs:boolean"/>
      <xs:element name="no-payload" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **VerifyMatch-RequestParams**, указанного в 16.34.

С.6.29 Параметры сообщения запроса ПМО БиоАПИ identifyMatch

```
<xs:complexType name="IdentifyMatch-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="maxFMRRequested" type="BioAPI-FMR"/>
    <xs:element name="processedBIR" type="BioAPI-INPUT-BIR"/>
    <xs:element name="population" type="BioAPI-IDENTIFY-POPULATION"/>
    <xs:element name="totalNumberOfTemplates" type="xs:unsignedInt"/>
    <xs:element name="binning" type="xs:boolean"/>
    <xs:element name="maxNumberOfResults" type="xs:unsignedInt"/>
    <xs:element name="timeout" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **IdentifyMatch-RequestParams**, указанного в 16.35.

С.6.30 Параметры сообщения запроса ПМО БиоАПИ enroll

```
<xs:complexType name="Enroll-RequestParams">
  <xs:sequence>
```

```
      <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
      <xs:element name="purpose" type="BioAPI-BIR-PURPOSE"/>
      <xs:element name="subtype" type="BioAPI-BIR-SUBTYPE"/>
      <xs:element name="outputFormat" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"
          minOccurs="0"/>
      <xs:element name="referenceTemplate" type="BioAPI-INPUT-BIR"/>
      <xs:element name="payload" type="BioAPI-DATA" minOccurs="0"/>
      <xs:element name="timeout" type="xs:int"/>
      <xs:element name="no-auditData" type="xs:boolean"/>
      <xs:element name="no-templateUuid" type="xs:boolean"/>
    </xs:sequence>
  </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **Enroll-RequestParams**, указанного в 16.36.

С.6.31 Параметры сообщения запроса ПМО БиоАПИ verify

```
<xs:complexType name="Verify-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="maxFMRRequested" type="BioAPI-FMR"/>
    <xs:element name="referenceTemplate" type="BioAPI-INPUT-BIR"/>
    <xs:element name="subtype" type="BioAPI-BIR-SUBTYPE"/>
    <xs:element name="timeout" type="xs:int"/>
    <xs:element name="no-adaptedBIR" type="xs:boolean"/>
    <xs:element name="no-fmrAchieved" type="xs:boolean"/>
    <xs:element name="no-payload" type="xs:boolean"/>
    <xs:element name="no-auditData" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **Verify-RequestParams**, указанного в 16.37.

С.6.32 Параметры сообщения запроса ПМО БиоАПИ identify

```
<xs:complexType name="Identify-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="maxFMRRequested" type="BioAPI-FMR"/>
    <xs:element name="subtype" type="BioAPI-BIR-SUBTYPE"/>
    <xs:element name="population" type="BioAPI-IDENTIFY-POPULATION"/>
    <xs:element name="totalNumberOfTemplates" type="xs:unsignedInt"/>
    <xs:element name="binning" type="xs:boolean"/>
    <xs:element name="maxNumberOfResults" type="xs:unsignedInt"/>
    <xs:element name="timeout" type="xs:int"/>
    <xs:element name="no-auditData" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Identify-RequestParams**, указанного в 16.38.

С.6.33 Параметры сообщения запроса ПМО БиоАПИ import

```
<xs:complexType name="Import-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="inputData" type="BioAPI-DATA"/>
    <xs:element name="inputFormat" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"/>
    <xs:element name="outputFormat" type="BioAPI-BIR-BIOMETRIC-DATA-FORMAT"
        minOccurs="0"/>
    <xs:element name="purpose" type="BioAPI-BIR-PURPOSE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Import-RequestParams**, указанного в 16.39.

С.6.34 Параметры сообщения запроса ПМО БиоАПИ presetIdentifyPopulation

```
<xs:complexType name="PresetIdentifyPopulation-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="population" type="BioAPI-IDENTIFY-POPULATION"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **PresetIdentifyPopulation-RequestParams**, указанного в 16.40.

С.6.35 Параметры сообщения запроса ПМО БиоАПИ transform

```
<xs:complexType name="Transform-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="operationUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Transform-RequestParams**, указанного в 16.41.

С.6.36 Параметры сообщения запроса ПМО БиоАПИ dbOpen

```
<xs:complexType name="DbOpen-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbUuid" type="BioAPI-UUID"/>
    <xs:element name="accessRequest" type="BioAPI-DB-ACCESS-TYPE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbOpen-RequestParams**, указанного в 16.42.

С.6.37 Параметры сообщения запроса ПМО БиоАПИ dbClose

```
<xs:complexType name="DbClose-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbHandle" type="BioAPI-DB-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbClose-RequestParams**, указанного в 16.43.

С.6.38 Параметры сообщения запроса ПМО БиоАПИ dbCreate

```
<xs:complexType name="DbCreate-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbUuid" type="BioAPI-UUID"/>
    <xs:element name="numberOfRecords" type="xs:unsignedInt"/>
    <xs:element name="accessRequest" type="BioAPI-DB-ACCESS-TYPE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbCreate-RequestParams**, указанного в 16.44.

С.6.39 Параметры сообщения запроса ПМО БиоАПИ dbDelete

```
<xs:complexType name="DbDelete-RequestParams">
  <xs:sequence>
```

```
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **DbDelete-RequestParams**, указанного в 16.45.

С.6.40 Параметры сообщения запроса ПМО БиоАПИ dbSetMarker

```
<xs:complexType name="DbSetMarker-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbHandle" type="BioAPI-DB-HANDLE"/>
    <xs:element name="keyValue" type="BioAPI-UUID"/>
    <xs:element name="markerHandle" type="BioAPI-DB-MARKER-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **DbSetMarker-RequestParams**, указанного в 16.46.

С.6.41 Параметры сообщения запроса ПМО БиоАПИ dbFreeMarker

```
<xs:complexType name="DbFreeMarker-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="markerHandle" type="BioAPI-DB-MARKER-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа АСН.1 **DbFreeMarker-RequestParams**, указанного в 16.47.

С.6.42 Параметры сообщения запроса ПМО БиоАПИ dbStoreBIR

```
<xs:complexType name="DbStoreBIR-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbHandle" type="BioAPI-DB-HANDLE"/>
    <xs:element name="birToStore" type="BioAPI-INPUT-BIR"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbStoreBIR-RequestParams**, указанного в 16.48.

С.6.43 Параметры сообщения запроса ПМО БиоАПИ dbGetBIR

```
<xs:complexType name="DbGetBIR-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbHandle" type="BioAPI-DB-HANDLE"/>
    <xs:element name="keyValue" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbGetBIR-RequestParams**, указанного в 16.49.

С.6.44 Параметры сообщения запроса ПМО БиоАПИ dbGetNextBIR

```
<xs:complexType name="DbGetNextBIR-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbHandle" type="BioAPI-DB-HANDLE"/>
    <xs:element name="markerHandle" type="BioAPI-DB-MARKER-HANDLE"/>
```

```
    </xs:sequence>
  </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **DbGetNextBIR-RequestParams**, указанного в 16.50.

С.6.45 Параметры сообщения запроса ПМО БиоАПИ dbDeleteBIR

```
<xs:complexType name="DbDeleteBIR-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="dbHandle" type="BioAPI-DB-HANDLE"/>
    <xs:element name="keyValue" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **DbDeleteBIR-RequestParams**, указанного в 16.51.

С.6.46 Параметры сообщения запроса ПМО БиоАПИ calibrateSensor

```
<xs:complexType name="CalibrateSensor-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="timeout" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **CalibrateSensor-RequestParams**, указанного в 16.52.

С.6.47 Параметры сообщения запроса ПМО БиоАПИ setPowerMode

```
<xs:complexType name="SetPowerMode-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="powerMode" type="BioAPI-POWER-MODE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **SetPowerMode-RequestParams**, указанного в 16.53.

С.6.48 Параметры сообщения запроса ПМО БиоАПИ setIndicatorStatus

```
<xs:complexType name="SetIndicatorStatus-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="indicatorStatus" type="BioAPI-INDICATOR-STATUS"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **SetIndicatorStatus-RequestParams**, указанного в 16.54.

С.6.49 Параметры сообщения запроса ПМО БиоАПИ getIndicatorStatus

```
<xs:complexType name="GetIndicatorStatus-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GetIndicatorStatus-RequestParams**, указанного в 16.55.

С.6.50 Параметры сообщения запроса ПМО БиоАПИ cancel

```
<xs:complexType name="Cancel-RequestParams">
  <xs:sequence>
    <xs:element name="originalBSPHandle" type="BioAPI-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Cancel-RequestParams**, указанного в 16.57.

С.6.51 Параметры сообщения запроса ПМО БиоАПИ registerBSP

```
<xs:complexType name="RegisterBSP-RequestParams">
  <xs:sequence>
    <xs:element name="bspSchema" type="BioAPI-BSP-SCHEMA"/>
    <xs:element name="update" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **RegisterBSP-RequestParams**, указанного в 16.59.

С.6.52 Параметры сообщения запроса ПМО БиоАПИ unregisterBSP

```
<xs:complexType name="UnregisterBSP-RequestParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **UnregisterBSP-RequestParams**, указанного в 16.60.

С.6.53 Параметры сообщения запроса ПМО БиоАПИ registerBFP

```
<xs:complexType name="RegisterBFP-RequestParams">
  <xs:sequence>
    <xs:element name="bfpSchema" type="BioAPI-BFP-SCHEMA"/>
    <xs:element name="update" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **RegisterBFP-RequestParams**, указанного в 16.61.

С.6.54 Параметры сообщения запроса ПМО БиоАПИ unregisterBFP

```
<xs:complexType name="UnregisterBFP-RequestParams">
  <xs:sequence>
    <xs:element name="bfpProductUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **UnregisterBFP-RequestParams**, указанного в 16.62.

С.7 Параметры сообщений ответа ПМО БиоАПИ

С.7.1 Параметры сообщения ответа ПМО БиоАПИ addMaster

```
<xs:complexType name="AddMaster-ResponseParams">
  <xs:sequence>
```

```
    <xs:element name="fwSchema" type="BioAPI-FRAMEWORK-SCHEMA"
      minOccurs="0"/>
    <xs:element name="bspSchema" type="BioAPI-BSP-SCHEMA"
      minOccurs="0" maxOccurs="4294967295"/>
    <xs:element name="bfpSchema" type="BioAPI-BFP-SCHEMA"
      minOccurs="0" maxOccurs="4294967295"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **AddMaster-ResponseParams**, указанного в 16.4.

С.7.2 Параметры сообщения ответа ПМО БиоАПИ deleteMaster

```
<xs:complexType name="DeleteMaster-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **DeleteMaster-ResponseParams**, указанного в 16.5.

С.7.3 Параметры сообщения ответа ПМО БиоАПИ bspLoad

```
<xs:complexType name="BSPLoad-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BSPLoad-ResponseParams**, указанного в 16.9.

С.7.4 Параметры сообщения ответа ПМО БиоАПИ bspUnload

```
<xs:complexType name="BSPUnload-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа ACH.1 **BSPUnload-ResponseParams**, указанного в 16.10.

С.7.5 Параметры сообщения ответа ПМО БиоАПИ queryUnits

```
<xs:complexType name="QueryUnits-ResponseParams">
  <xs:sequence>
    <xs:element name="unitSchemas">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="unitSchema" type="BioAPI-UNIT-SCHEMA"
            minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **QueryUnits-ResponseParams**, указанного в 16.11.

С.7.6 Параметры сообщения ответа ПМО БиоАПИ queryBFPs

```
<xs:complexType name="QueryBFPs-ResponseParams">
  <xs:sequence>
    <xs:element name="bfps">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="bfp" type="BioAPI-BFP-LIST-ELEMENT"
            minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **QueryBFPs-ResponseParams**, указанного в 16.12.

С.7.7 Параметры сообщения ответа ПМО БиоАПИ bspAttach

```
<xs:complexType name="BSPAttach-ResponseParams">
  <xs:sequence>
    <xs:element name="newOriginalBSPHandle" type="BioAPI-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BSPAttach-ResponseParams**, указанного в 16.13.

С.7.8 Параметры сообщения ответа ПМО БиоАПИ bspDetach

```
<xs:complexType name="BSPDetach-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BSPDetach-ResponseParams**, указанного в 16.14.

С.7.9 Параметры сообщения ответа ПМО БиоАПИ enableUnitEvents

```
<xs:complexType name="EnableUnitEvents-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **EnableUnitEvents-ResponseParams**, указанного в 16.15.

C.7.10 Параметры сообщения ответа ПМО БиоАПИ enableEventNotifications

```
<xs:complexType name="EnableEventNotifications-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **EnableEventNotifications-ResponseParams**, указанного в 16.16.

C.7.11 Параметры сообщения ответа ПМО БиоАПИ controlUnit

```
<xs:complexType name="ControlUnit-ResponseParams">
  <xs:sequence>
    <xs:element name="outputData" type="BioAPI-DATA"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **ControlUnit-ResponseParams**, указанного в 16.17.

C.7.12 Параметры сообщения ответа ПМО БиоАПИ control

```
<xs:complexType name="Control-ResponseParams">
  <xs:sequence>
    <xs:element name="outputData" type="BioAPI-DATA"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Control-ResponseParams**, указанного в 16.18.

C.7.13 Параметры сообщения ответа ПМО БиоАПИ freeBIRHandle

```
<xs:complexType name="FreeBIRHandle-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **FreeBIRHandle-ResponseParams**, указанного в 16.19.

C.7.14 Параметры сообщения ответа ПМО БиоАПИ getBIRFromHandle

```
<xs:complexType name="GetBIRFromHandle-ResponseParams">
  <xs:sequence>
    <xs:element name="bir" type="BioAPI-BIR"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GetBIRFromHandle-ResponseParams**, указанного в 16.20.

C.7.15 Параметры сообщения ответа ПМО БиоАПИ getHeaderFromHandle

```
<xs:complexType name="GetHeaderFromHandle-ResponseParams">
  <xs:sequence>
    <xs:element name="header" type="BioAPI-BIR-HEADER"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GetHeaderFromHandle-ResponseParams**, указанного в 16.21.

С.7.16 Параметры сообщения ответа ПМО БиоАПИ subscribeToGUIEvents

```
<xs:complexType name="SubscribeToGUIEvents-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **SubscribeToGUIEvents-ResponseParams**, указанного в 16.22.

С.7.17 Параметры сообщения ответа ПМО БиоАПИ unsubscribeFromGUIEvents

```
<xs:complexType name="UnsubscribeFromGUIEvents-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **UnsubscribeFromGUIEvents-ResponseParams**, указанного в 16.23.

С.7.18 Параметры сообщения ответа ПМО БиоАПИ queryGUIEventSubscriptions

```
<xs:complexType name="QueryGUIEventSubscriptions-ResponseParams">
  <xs:sequence>
    <xs:element name="guiEventSubscriptions">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="subscription" type="BioAPI-BFP-LIST-ELEMENT"
              minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа ACH.1 **QueryGUIEventSubscriptions-**ResponseParams, указанного в 16.24.

С.7.19 Параметры сообщения ответа ПМО БиоАПИ notifyGUISelectEvent

```
<xs:complexType name="NotifyGUISelectEvent-ResponseParams">
  <xs:sequence>
    <xs:element name="selectedInstances" type="BioAPI-BIR-SUBTYPE-MASK"/>
    <xs:element name="response" type="BioAPI-GUI-RESPONSE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **NotifyGUISelectEvent-ResponseParams**, указанного в 16.25.

С.7.20 Параметры сообщения ответа ПМО БиоАПИ notifyGUIStateEvent

```
<xs:complexType name="NotifyGUIStateEvent-ResponseParams">
  <xs:sequence>
    <xs:element name="response" type="BioAPI-GUI-RESPONSE"/>
    <xs:element name="enrollSampleIndexToRecapture" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **NotifyGUIStateEvent-ResponseParams**, указанного в 16.26.

С.7.21 Параметры сообщения ответа ПМО БиоАПИ notifyGUIProgressEvent

```
<xs:complexType name="NotifyGUIProgressEvent-ResponseParams">
  <xs:sequence>
```

```
    <xs:element name="response" type="BioAPI-GUI-RESPONSE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **NotifyGUIProgressEvent-ResponseParams**, указанного в 16.27.

C.7.22 Параметры сообщения ответа ПМО БиоАПИ redirectGUIEvents

```
<xs:complexType name="RedirectGUIEvents-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **RedirectGUIEvents-ResponseParams**, указанного в 16.28.

C.7.23 Параметры сообщения ответа ПМО БиоАПИ unredirectGUIEvents

```
<xs:complexType name="UnredirectGUIEvents-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **UnredirectGUIEvents-ResponseParams**, указанного в 16.29.

C.7.24 Параметры сообщения ответа ПМО БиоАПИ capture

```
<xs:complexType name="Capture-ResponseParams">
  <xs:sequence>
    <xs:element name="capturedBIR" type="BioAPI-BIR-HANDLE"/>
    <xs:element name="auditData" type="BioAPI-BIR-HANDLE" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **Capture-ResponseParams**, указанного в 16.30.

С.7.25 Параметры сообщения ответа ПМО БиоАПИ createTemplate

```
<xs:complexType name="CreateTemplate-ResponseParams">
  <xs:sequence>
    <xs:element name="newTemplate" type="BioAPI-BIR-HANDLE"/>
    <xs:element name="templateUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **CreateTemplate-ResponseParams**, указанного в 16.31.

С.7.26 Параметры сообщения ответа ПМО БиоАПИ process

```
<xs:complexType name="Process-ResponseParams">
  <xs:sequence>
    <xs:element name="processedBIR" type="BioAPI-BIR-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **Process-ResponseParams**, указанного в 16.32.

С.7.27 Параметры сообщения ответа ПМО БиоАПИ processWithAuxBIR

```
<xs:complexType name="ProcessWithAuxBIR-ResponseParams">
  <xs:sequence>
    <xs:element name="processedBIR" type="BioAPI-BIR-HANDLE"/>
```

```
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **ProcessWithAuxBIR-ResponseParams**, указанного в 16.33.

С.7.28 Параметры сообщения ответа ПМО БиоАПИ verifyMatch

```
<xs:complexType name="VerifyMatch-ResponseParams">
  <xs:sequence>
    <xs:element name="adaptedBIR" type="BioAPI-BIR-HANDLE" minOccurs="0"/>
    <xs:element name="result" type="xs:boolean"/>
    <xs:element name="fmrAchieved" type="BioAPI-FMR" minOccurs="0"/>
    <xs:element name="payload" type="BioAPI-DATA" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **VerifyMatch-ResponseParams**, указанного в 16.34.

С.7.29 Параметры сообщения ответа ПМО БиоАПИ identifyMatch

```
<xs:complexType name="IdentifyMatch-ResponseParams">
  <xs:sequence>
    <xs:element name="candidates">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="candidate" type="BioAPI-CANDIDATE"
              minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **IdentifyMatch-ResponseParams**, указанного в 16.35.

С.7.30 Параметры сообщения ответа ПМО БиоАПИ enroll

```
<xs:complexType name="Enroll-ResponseParams">
  <xs:sequence>
    <xs:element name="newTemplate" type="BioAPI-BIR-HANDLE"/>
    <xs:element name="adaptedBIR" type="BioAPI-BIR-HANDLE" minOccurs="0"/>
    <xs:element name="templateUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Enroll-ResponseParams**, указанного в 16.36.

С.7.31 Параметры сообщения ответа ПМО БиоАПИ verify

```
<xs:complexType name="Verify-ResponseParams">
  <xs:sequence>
    <xs:element name="adaptedBIR" type="BioAPI-BIR-HANDLE" minOccurs="0"/>
    <xs:element name="result" type="xs:boolean"/>
    <xs:element name="fmrAchieved" type="BioAPI-FMR" minOccurs="0"/>
    <xs:element name="payload" type="BioAPI-DATA" minOccurs="0"/>
    <xs:element name="auditData" type="BioAPI-BIR-HANDLE" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Verify-ResponseParams**, указанного в 16.37.

С.7.32 Параметры сообщения ответа ПМО БиоАПИ identify

```
<xs:complexType name="Identify-ResponseParams">
  <xs:sequence>
    <xs:element name="candidates">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="candidate" type="BioAPI-CANDIDATE"
              minOccurs="0" maxOccurs="4294967295"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="auditData" type="BioAPI-BIR-HANDLE" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Identify-ResponseParams**, указанного в 16.38.

С.7.33 Параметры сообщения ответа ПМО БиоАПИ import

```
<xs:complexType name="Import-ResponseParams">
  <xs:sequence>
    <xs:element name="constructedBIR" type="BioAPI-BIR-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Import-ResponseParams**, указанного в 16.39.

С.7.34 Параметры сообщения ответа ПМО БиоАПИ presetIdentifyPopulation

```
<xs:complexType name="PresetIdentifyPopulation-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения АСН.1 типа **PresetIdentifyPopulation-ResponseParams**, указанного в 16.40.

С.7.35 Параметры сообщения ответа ПМО БиоАПИ transform

```
<xs:complexType name="Transform-ResponseParams">
  <xs:sequence>
    <xs:element name="outputBIRs" type="BioAPI-OUTPUT-BIR" minOccurs="0"
    maxOccurs="4294967295"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **Transform-ResponseParams**, указанного в 16.41.

С.7.36 Параметры сообщения ответа ПМО БиоАПИ dbOpen

```
<xs:complexType name="DbOpen-ResponseParams">
  <xs:sequence>
    <xs:element name="dbHandle" type="BioAPI-DB-HANDLE"/>
    <xs:element name="markerHandle" type="BioAPI-DB-MARKER-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbOpen-ResponseParams**, указанного в 16.42.

С.7.37 Параметры сообщения ответа ПМО БиоАПИ dbClose

```
<xs:complexType name="DbClose-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа АСН.1 **DbClose-ResponseParams**, указанного в 16.43.

С.7.38 Параметры сообщения ответа ПМО БиоАПИ dbCreate

```
<xs:complexType name="DbCreate-ResponseParams">
  <xs:sequence>
    <xs:element name="dbHandle" type="BioAPI-DB-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbCreate-ResponseParams**, указанного в 16.44.

С.7.39 Параметры сообщения ответа ПМО БиоАПИ dbDelete

```
<xs:complexType name="DbDelete-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbDelete-ResponseParams**, указанного в 16.45.

С.7.40 Параметры сообщения ответа ПМО БиоАПИ dbSetMarker

```
<xs:complexType name="DbSetMarker-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbSetMarker-ResponseParams**, указанного в 16.46.

С.7.41 Параметры сообщения ответа ПМО БиоАПИ dbFreeMarker

```
<xs:complexType name="DbFreeMarker-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbFreeMarker-ResponseParams**, указанного в 16.47.

С.7.42 Параметры сообщения ответа ПМО БиоАПИ dbStoreBIR

```
<xs:complexType name="DbStoreBIR-ResponseParams">
  <xs:sequence>
    <xs:element name="birUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbStoreBIR-ResponseParams**, указанного в 16.48.

С.7.43 Параметры сообщения ответа ПМО БиоАПИ dbGetBIR

```
<xs:complexType name="DbGetBIR-ResponseParams">
  <xs:sequence>
    <xs:element name="retrievedBIR" type="BioAPI-BIR-HANDLE"/>
    <xs:element name="markerHandle" type="BioAPI-DB-MARKER-HANDLE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **DbGetBIR-ResponseParams**, указанного в 16.49.

С.7.44 Параметры сообщения ответа ПМО БиоАПИ dbGetNextBIR

```
<xs:complexType name="DbGetNextBIR-ResponseParams">
  <xs:sequence>
    <xs:element name="retrievedBIR" type="BioAPI-BIR-HANDLE"/>
    <xs:element name="birUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **DbGetNextBIR-ResponseParams**, указанного в 16.50.

С.7.45 Параметры сообщения ответа ПМО БиоАПИ dbDeleteBIR

```
<xs:complexType name="DbDeleteBIR-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **DbDeleteBIR-ResponseParams**, указанного в 16.51.

С.7.46 Параметры сообщения ответа ПМО БиоАПИ calibrateSensor

```
<xs:complexType name="CalibrateSensor-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **CalibrateSensor-ResponseParams**, указанного в 16.52.

С.7.47 Параметры сообщения ответа ПМО БиоАПИ setPowerMode

```
<xs:complexType name="SetPowerMode-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **SetPowerMode-ResponseParams**, указанного в 16.53.

С.7.48 Параметры сообщения ответа ПМО БиоАПИ setIndicatorStatus

```
<xs:complexType name="SetIndicatorStatus-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **SetIndicatorStatus-ResponseParams**, указанного в 16.54.

С.7.49 Параметры сообщения ответа ПМО БиоАПИ getIndicatorStatus

```
<xs:complexType name="GetIndicatorStatus-ResponseParams">
  <xs:sequence>
    <xs:element name="indicatorStatus" type="BioAPI-INDICATOR-STATUS"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GetIndicatorStatus-ResponseParams**, указанного в 16.55.

С.7.50 Параметры сообщения ответа ПМО БиоАПИ cancel

```
<xs:complexType name="Cancel-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **Cancel-ResponseParams**, указанного в 16.57.

С.7.51 Параметры сообщения ответа ПМО БиоАПИ registerBSP

```
<xs:complexType name="RegisterBSP-ResponseParams"/>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения ACH.1 типа **RegisterBSP-ResponseParams**, указанного в 16.59.

C.7.52 Параметры сообщения ответа ПМО БиоАПИ unregisterBSP

<xs:complexType name="UnregisterBSP-ResponseParams"/>

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **UnregisterBSP-ResponseParams**, указанного в 16.60.

C.7.53 Параметры сообщения ответа ПМО БиоАПИ registerBFP

<xs:complexType name="RegisterBFP-ResponseParams"/>

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **RegisterBFP-ResponseParams**, указанного в 16.61.

C.7.54 Параметры сообщения ответа ПМО БиоАПИ unregisterBFP

<xs:complexType name="UnregisterBFP-ResponseParams"/>

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения ACH.1 типа **UnregisterBFP-ResponseParams**, указанного в 16.62.

C.8 Параметры сообщений уведомления ПМО БиоАПИ

C.8.1 Параметры сообщения уведомления ПМО БиоАПИ masterDeletionEvent

<xs:complexType name="MasterDeletionEvent-NotificationParams"/>

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **MasterDeletionEvent-NotificationParams**, указанного в 16.3.

С.8.2 Параметры сообщения уведомления ПМО БиоАПИ unitEvent

```
<xs:complexType name="UnitEvent-NotificationParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="unitSchema" type="BioAPI-UNIT-SCHEMA" minOccurs="0"/>
    <xs:element name="unitEventType" type="BioAPI-UNIT-EVENT-TYPE"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **UnitEvent-NotificationParams**, указанного в 17.1.

С.8.3 Параметры сообщения уведомления ПМО БиоАПИ guiSelectEvent

```
<xs:complexType name="GUISelectEvent-NotificationParams">
  <xs:sequence>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"
        minOccurs="0"/>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="originalBSPHandle" minOccurs="0"/>
    <xs:element name="enrollType" type="BioAPI-GUI-ENROLL-TYPE"/>
    <xs:element name="operation" type="BioAPI-GUI-OPERATION"/>
    <xs:element name="moment" type="BioAPI-GUI-MOMENT"/>
    <xs:element name="resultCode" type="BioAPI-RETURN"/>
    <xs:element name="maxNumEnrollSamples" type="xs:unsignedInt"/>
    <xs:element name="selectableInstances" type="BioAPI-BIR-SUBTYPE-MASK"/>
    <xs:element name="capturedInstances" type="BioAPI-BIR-SUBTYPE-MASK"/>
    <xs:element name="text" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование

абстрактного значения типа АСН.1 **GUISelectEvent-NotificationParams**, указанного в 17.2.

С.8.4 Параметры сообщения уведомления ПМО БиоАПИ guiStateEvent

```
<xs:complexType name="GUIStateEvent-NotificationParams">
  <xs:sequence>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"
        minOccurs="0"/>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="originalBSPHandle" minOccurs="0"/>
    <xs:element name="operation" type="BioAPI-GUI-OPERATION"/>
    <xs:element name="suboperation" type="BioAPI-GUI-SUBOPERATION"/>
    <xs:element name="purpose" type="BioAPI-BIR-PURPOSE"/>
    <xs:element name="moment" type="BioAPI-GUI-MOMENT"/>
    <xs:element name="resultCode" type="BioAPI-RETURN"/>
    <xs:element name="enrollSampleIndex" type="xs:int"/>
    <xs:element name="bitmaps" type="BioAPI-GUI-BITMAP-ARRAY" minOccurs="0"/>
    <xs:element name="text" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа АСН.1 **GUIStateEvent-NotificationParams**, указанного в 17.3.

С.8.5 Параметры сообщения уведомления ПМО БиоАПИ guiProgressEvent

```
<xs:complexType name="GUIProgressEvent-NotificationParams">
  <xs:sequence>
    <xs:element name="guiEventSubscriptionUuid" type="BioAPI-UUID"
        minOccurs="0"/>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
    <xs:element name="unitID" type="BioAPI-UNIT-ID"/>
    <xs:element name="originalBSPHandle" minOccurs="0"/>
    <xs:element name="operation" type="BioAPI-GUI-OPERATION"/>
```

```
      <xs:element name="suboperation" type="BioAPI-GUI-SUBOPERATION"/>
      <xs:element name="purpose" type="BioAPI-BIR-PURPOSE"/>
      <xs:element name="moment" type="BioAPI-GUI-MOMENT"/>
      <xs:element name="suboperationProgress" type="xs:unsignedByte"/>
      <xs:element name="bitmaps" type="BioAPI-GUI-BITMAP-ARRAY" minOccurs="0"/>
      <xs:element name="text" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GUIProgressEvent-NotificationParams**, указанного в 17.4.

С.8.6 Параметры сообщения уведомления ПМО БиоАПИ bspRegistrationEvent

```
<xs:complexType name="BSPRegistrationEvent-NotificationParams">
  <xs:sequence>
    <xs:element name="bspSchema" type="BioAPI-BSP-SCHEMA"/>
    <xs:element name="update" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BSPRegistrationEvent-NotificationParams**, указанного в 16.59.

С.8.7 Параметры сообщения уведомления ПМО БиоАПИ bspUnregistrationEvent

```
<xs:complexType name="BSPUnregistrationEvent-NotificationParams">
  <xs:sequence>
    <xs:element name="bspProductUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BSPUnregistrationEvent-NotificationParams**, указанного в 16.60.

С.8.8 Параметры сообщения уведомления ПМО БиоАПИ bfpRegistrationEvent

```
<xs:complexType name="BFPRegistrationEvent-NotificationParams">
  <xs:sequence>
    <xs:element name="bfpSchema" type="BioAPI-BFP-SCHEMA"/>
    <xs:element name="update" type="xs:boolean"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BFPRegistrationEvent-NotificationParams**, указанного в 16.61.

С.8.9 Параметры сообщения уведомления ПМО БиоАПИ bfpUnregistrationEvent

```
<xs:complexType name="BFPUnregistrationEvent-NotificationParams">
  <xs:sequence>
    <xs:element name="bfpProductUuid" type="BioAPI-UUID"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **BFPUnregistrationEvent-NotificationParams**, указанного в 16.62.

С.9 Параметры сообщений подтверждения ПМО БиоАПИ

С.9.1 Параметры сообщения подтверждения ПМО БиоАПИ guiSelectEvent

```
<xs:complexType name="GUISelectEvent-AcknowledgementParams">
  <xs:sequence>
    <xs:element name="response" type="BioAPI-GUI-RESPONSE"/>
    <xs:element name="selectedInstances" type="BioAPI-BIR-SUBTYPE-MASK"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GUISelectEvent-AcknowledgementParams**, указанного в 17.2.

С.9.2 Параметры сообщения подтверждения ПМО БиоАПИ guiStateEvent

```
<xs:complexType name="GUIStateEvent-AcknowledgementParams">
  <xs:sequence>
    <xs:element name="response" type="BioAPI-GUI-RESPONSE"/>
    <xs:element name="enrollSampleIndexToRecapture" type="xs:int"/>
  </xs:sequence>
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GUIStateEvent-AcknowledgementParams**, указанного в 17.3.

С.9.3 Параметры сообщения подтверждения ПМО БиоАПИ guiProgressEvent

```
<xs:complexType name="GUIProgressEvent-AcknowledgementParams">
  <xs:sequence>
    <xs:element name="response" type="BioAPI-GUI-RESPONSE"/>
  </xs:sequence>
```

```
</xs:complexType>
```

Семантика и использование данного типа XSD определены путем интерпретации элемента данного типа как EXTENDED-XER кодирование абстрактного значения типа ACH.1 **GUIProgressEvent-AcknowledgementParams**, указанного в 17.4.

С.10 Закрытие схемы

```
</xs:schema>
```

С.11 Пример

В таблице С.1 приведен пример последовательности сообщений, обмен которых происходит в привязке SOAP /HTTP.

Таблица С.1 – Пример обмена сообщениями ПМО БиоАПИ в SOAP /HTTP привязке ПМО БиоАПИа

HTTP сообщение запроса или ответа	Действие
POST /BIPRequest/addMaster HTTP/1.1 **Host: slave.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Request"** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:request xmlns:b="oid:/BIP">** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000001</b:requestId>** **<b:addMaster>** **<b:bipVersion>** **<b:major>1</b:major><b:minor>0</b:minor>** **</b:bipVersion>** **</b:addMaster>** **</b:request>** **</s:Body>** **</s:Envelope>**	1а Главная конечная точка ПМО БиоАПИ создает связь ПМО БиоАПИ и отправляет запрос **addMaster** в новую второстепенную точку

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 5151** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:response xmlns:b="oid:/BIP">** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000001</b:requestId>** **<b:addMaster>** **<b:fwSchema>** **<b:fwProductUuid>** **871c33d1-50a0-9de1-a502-06df190aae61** **</b:fwProductUuid>** **<b:description>** **BioAPI 2.0 Framework v15.3c** **</b:description>** **<b:specVersion>** **<b:major>2</b:major><b:minor>0</b:minor>** **</b:specVersion>** **<b:productVersion>15.3c</b:productVersion>** **<b:vendor>Hksjdgsg Wueryz, Inc.</b:vendor>** **<b:hostingEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:hostingEndpointIRI>** **</b:fwSchema>**	1b Второстепенная конечная точка создает ответ **addMaster**, содержащий информацию из ее реестра компонентов (схемы структуры и схем ПБУ), и отправляет его в главную второстепенную точку

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
<b:bspSchema> <b:bspProductUuid> 81a67810-19e3-45a7-3518-7165ee4a5ac4 </b:bspProductUuid> <b:description> Fingerprint BSP </b:description> <b:specVersion> <b:major>2</b:major><b:minor>0</b:minor> </b:specVersion> <b:productVersion>6.5</b:productVersion> <b:vendor>Kjfahgtrtwt Eggdad, Inc.</b:vendor> <b:supportedFormats> <b:format> <b:formatOwner>151</b:formatOwner> <b:formatType>5646</b:formatType> </b:format> <b:format> <b:formatOwner>65</b:formatOwner> <b:formatType>5</b:formatType> </b:format> </b:supportedFormats> <b:factorsMask>fingerprint</b:factorsMask> <b:operations> process createTemplate verifyMatch identifyMatch presetIdentifyPopulation databaseOperations queryUnits </b:operations> <b:options> qualityIntermediate qualityProcessed payload birSign birEncrypt templateUpdate adaptation </b:options>	

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
<b:payloadPolicy>512</b:payloadPolicy> <b:maxPayloadSize>1000000000</b:maxPayloadSize> <b:defaultVerifyTimeout>10</b:defaultVerifyTimeout> <b:defaultIdentifyTimeout>10</b:defaultIdentifyTimeout> <b:defaultCaptureTimeout>10</b:defaultCaptureTimeout> <b:defaultEnrollTimeout>10</b:defaultEnrollTimeout> <b:defaultCalibrateTimeout>10</b:defaultCalibrateTimeout> <b:maxBSPDbSize>1000000</b:maxBSPDbSize> <b:maxIdentify>1</b:maxIdentify> <b:hostingEndpointIRI> http://slave.example.org/BIP/kjfgsfa4146 </b:hostingEndpointIRI> </b:bspSchema> <b:bspSchema> <b:bspProductUuid> 7711e4a4-7a15-ea54-4616-9734374ea422 </b:bspProductUuid> <b:description> Iris BSP </b:description> <b:specVersion> <b:major>2</b:major><b:minor>0</b:minor> </b:specVersion> <b:productVersion>7.1g</b:productVersion> <b:vendor>lodfghrsrgf Uyhsgh, Inc.</b:vendor> <b:supportedFormats> <b:format> <b:formatOwner>677</b:formatOwner> <b:formatType>16</b:formatType> </b:format> <b:format> <b:formatOwner>241</b:formatOwner> <b:formatType>41</b:formatType> </b:format>	

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
</b:supportedFormats> <b:factorsMask>iris</b:factorsMask> <b:operations> process createTemplate verifyMatch identifyMatch presetIdentifyPopulation databaseOperations queryUnits </b:operations> <b:options> qualityIntermediate qualityProcessed payload templateUpdate </b:options> <b:payloadPolicy>100</b:payloadPolicy> <b:maxPayloadSize>40000</b:maxPayloadSize> <b:defaultVerifyTimeout>10</b:defaultVerifyTimeout> <b:defaultIdentifyTimeout>10</b:defaultIdentifyTimeout> <b:defaultCaptureTimeout>10</b:defaultCaptureTimeout> <b:defaultEnrollTimeout>10</b:defaultEnrollTimeout> <b:defaultCalibrateTimeout>10</b:defaultCalibrateTimeout> <b:maxBSPDbSize>1000000</b:maxBSPDbSize> <b:maxIdentify>1</b:maxIdentify> <b:hostingEndpointIRI> http://slave.example.org/BIP/kjfgsfa4146 </b:hostingEndpointIRI> </b:bspSchema> </b:addMaster> <b:returnValue>0</b:returnValue> </b:response> </s:Body> </s:Envelope>	

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
POST /BIPRequest/bspLoad HTTP/1.1 **Host: slave.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Request"** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:request xmlns:b="oid:/BIP">** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000002</b:requestId>** **<b:bspLoad>** **<b:bspProductUuid>** **81a67810-19e3-45a7-3518-7165ee4a5ac4** **</b:bspProductUuid>** **<b:unitEventSubscription>true</b:unitEventSubscription>** **</b:bspLoad>** **</b:request>** **</s:Body>** **</s:Envelope>**	2а. Главная второстепенная точка отправляет запрос **bspLoad** во второстепенную конечную точку

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 314** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:response xmlns:b="oid:/BIP">** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000002</b:requestId>** **<b:bspLoad/>** **<b:returnValue>0</b:returnValue>** **</b:response>** **</s:Body>** **</s:Envelope>** **POST /BIPNotification/unitEvent HTTP/1.1** **Host: master.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Notification"**	2b Второстепенная конечная точка создает ответ **bspLoad**

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
<?xml version="1.0" encoding="UTF-8"?> <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"> <s:Body> <b:notification xmlns:b="oid:/BIP"> <b:slaveEndpointIRI> http://slave.example.org/BIP/kjfgsfa4146 </b:slaveEndpointIRI> <b:masterEndpointIRI> http://master.example.org/BIP/fthjhuh5521 </b:masterEndpointIRI> <b:linkNumber>197114367</b:linkNumber> <b:notificationId>500000001</b:notificationId> <b:unitEvent> <b:bspProductUuid> 81a67810-19e3-45a7-3518-7165ee4a5ac4 </b:bspProductUuid> <b:unitID>1</b:unitID> <b:unitSchema> <b:bspProductUuid> 81a67810-19e3-45a7-3518-7165ee4a5ac4 </b:bspProductUuid> <b:unitManagerProductUuid> 81a67810-19e3-45a7-3518-7165ee4a5ac4 </b:unitManagerProductUuid> <b:unitID>1</b:unitID> <b:category>comparisonAlgorithm</b:category> <b:unitProperties> 81a67810-19e3-45a7-3518-7165ee4a5ac4 </b:unitProperties><b:vendorInformation> Kjsdfhasdf lsgdfgdgdg, Inc. </b:vendorInformation> <b:supportedUnitEvents>	

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
insert remove </b:supportedUnitEvents> <b:propertyUuid> 81a67810-19e3-45a7-3518-7165ee4a5ac4 </b:propertyUuid> <b:property> UjBsR09EbGhjZ0dT </b:property> <b:hardwareVersion>1.0</b:hardwareVersion> <b:firmwareVersion>1.1</b:firmwareVersion> <b:softwareVersion>5.2</b:softwareVersion> <b:hardwareSerialNumber>12132323</b:hardwareSerialNumber> <b:authenticatedHardware>true</b:authenticatedHardware> <b:maxBSPDbSize>1000000</b:maxBSPDbSize> <b:maxIdentify>1</b:maxIdentify> </b:unitSchema> <b:unitEventType>insert</b:unitEventType> </b:unitEvent> </b:notification> </s:Body> </s:Envelope>	

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Length: 0**	3b Главная конечная точка не отправляет подтверждения в ответ на уведомление **unitEvent**, но необходим ответ HTTP

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
POST /BIPRequest/bspAttach HTTP/1.1 **Host: slave.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Request"** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:request xmlns:b="oid:/BIP">** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000003</b:requestId>** **<b:bspAttach>** **<b:bspProductUuid>** **81a67810-19e3-45a7-3518-7165ee4a5ac4** **</b:bspProductUuid>** **<b:version>** **<b:major>2</b:major><b:minor>0</b:minor>** **</b:version>** **<b:units>** **<b:unit>** **<b:category>processingAlgorithm</b:category>** **<b:unitID>6</b:unitID>** **</b:unit>** **<b:unit>**	4а Главная конечная точка отправляет запрос **bspAttach** во второстепенную конечную точку

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
<b:category>comparisonAlgorithm</b:category> <b:unitID>1</b:unitID> </b:unit> </b:units> </b:bspAttach> </b:request> </s:Body> </s:Envelope>	

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 314** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:response xmlns:b="oid:/BIP">** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000003</b:requestId>** **<b:bspAttach>** **<b:newOriginalBSPHandle>716</b:newOriginalBSPHandle>** **</b:bspAttach>** **<b:returnValue>0</b:returnValue>** **</b:response>** **</s:Body>** **</s:Envelope>**	4b Второстепенная конечная точка создает ответ **bspAttach**

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
POST /BIPRequest/createTemplate HTTP/1.1 **Host: slave.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Request"** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:request xmlns:b="oid:/BIP">** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000004</b:requestId>** **<b:createTemplate>** **<b:originalBSPHandle>716</b:originalBSPHandle>** **<b:capturedBIR>** **<b:bir patronFormatOwner="257" patronFormatType="14">** **<x:bir xmlns:x="urn:oid:1.1.19785.0.257.1.8.0">** **<x:version major="0" minor="0"/>** **<x:cbeff-version major="2" minor="0"/>** **<x:bir-info** **integrity="true"** **creation-date="20040302T150315Z"** **not-valid-after="20040302T4415Z">** **<x:index>** **1Aa873ab3auE61cCa91723d6P==** **</x:index>** **</x:bir-info>**	5а Главная конечная точка отправляет запрос **createTemplate** во второстепенную конечную точку. Запрос содержит полную ЗБИ в XML формате ведущей организации

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
<x:bdb-info format-owner="51" format-type="8" encryption="true" type="iris" subtype="left" level="intermediate" product-owner="16" product-type="2" purpose="enroll" quality="100"/> <x:sb-info format-owner="51" format-type="8"/> <x:bdb> 1Aa83sdfsk82uw7sfjsfssga8977896254sdf724 uEKhsjgfhJHGskdfkhghsgdf77823skldDfsd61c CewiuiUisdgfkjHjdshlsd65gkdgjdga93d6P== </x:bdb> <x:sb> 1Aa873ab3auE61cCa9kjkfdg7878skjfsdf gjfdigjdigsdg1723d6P== </x:sb> </x:bir> </b:bir> </b:capturedBIR> </b:createTemplate> </b:request> </s:Body> </s:Envelope>	

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 314** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:response xmlns:b="oid:/BIP">** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000004</b:requestId>** **<b:createTemplate>** **<b:newTemplate>1832018</b:newTemplate>** **<b:templateUuid>** **11763242-a786-4d41-87a3-9774ee4a51c4** **</b:templateUuid>** **</b:createTemplate>** **<b:returnValue>0</b:returnValue>** **</b:response>** **</s:Body>** **</s:Envelope>**	5b Второстепенная конечная точка создает ответ **createTemplate**

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
POST /BIPRequest/verifyMatch HTTP/1.1 **Host: slave.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Request"** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:request xmlns:b="oid:/BIP">** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000005</b:requestId>** **<b:verifyMatch>** **<b:originalBSPHandle>716</b:originalBSPHandle>** **<b:maxFMRRequested>42</b:maxFMRRequested>** **<b:processedBIR>** **<b:birInBSP>5867461</b:birInBSP>** **</b:processedBIR>** **<b:referenceTemplate>** **<b:birInBSP>66241</b:birInBSP>** **</b:referenceTemplate>** **<b:no-adaptedBIR>true</b:no-adaptedBIR>** **<b:no-fmrAchieved>true</b:no-fmrAchieved>** **<b:no-payload>true</b:no-payload>** **</b:verifyMatch>** **</b:request>** **</s:Body>** **</s:Envelope>**	6а Главная конечная точка отправляет запрос **verifyMatch** во второстепенную

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 314** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:response xmlns:b="oid:/BIP">** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000005</b:requestId>** **<b:verifyMatch>** **<b:result>false</b:result>** **<b:fmrAchieved>497840</b:fmrAchieved>** **</b:verifyMatch>** **<b:returnValue>0</b:returnValue>** **</b:response>** **</s:Body>** **</s:Envelope>**	6b Второстепенная конечная точка создает ответ **verifyMatch**

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
POST /BIPRequest/bspDetach HTTP/1.1 **Host: slave.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Request"** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:request xmlns:b="oid:/BIP">** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000006</b:requestId>** **<b:bspDetach>** **<b:originalBSPHandle>716</b:originalBSPHandle>** **</b:bspDetach>** **</b:request>** **</s:Body>** **</s:Envelope>**	7а Главная конечная точка отправляет запрос **bspDetach** во второстепенную конечную точку

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 314** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:response xmlns:b="oid:/BIP">** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000006</b:requestId>** **<b:bspDetach/>** **<b:returnValue>0</b:returnValue>** **</b:response>** **</s:Body>** **</s:Envelope>**	7b Второстепенная конечная точка создает ответ **bspDetach**

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
POST /BIPRequest/bspUnload HTTP/1.1 **Host: slave.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Request"** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:request xmlns:b="oid:/BIP">** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000007</b:requestId>** **<b:bspUnload>** **<b:bspProductUuid>** **81a67810-19e3-45a7-3518-7165ee4a5ac4** **</b:bspProductUuid>** **<b:unitEventSubscription>true</b:unitEventSubscription>** **</b:bspUnload>** **</b:request>** **</s:Body>** **</s:Envelope>**	8a Главная конечная точка отправляет запрос **bspUnload** во второстепенную конечную точку

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 314** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:response xmlns:b="oid:/BIP">** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000007</b:requestId>** **<b:bspUnload/>** **<b:returnValue>0</b:returnValue>** **</b:response>** **</s:Body>** **</s:Envelope>**	8b Второстепенная конечная точка создает ответ **bspUnload**

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
POST /BIPRequest/deleteMaster HTTP/1.1 **Host: slave.example.org** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 415** **SOAPAction: "oid:/BIP/Request"** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:request xmlns:b="oid:/BIP">** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000008</b:requestId>** **<b:deleteMaster/>** **</b:request>** **</s:Body>** **</s:Envelope>**	9а Главная конечная точка отправляет запрос **deleteMaster** во второстепенную конечную точку для объявления о намерении разрушения связи ПМО БиоАПИ

Продолжение таблицы С.1

HTTP сообщение запроса или ответа	Действие
HTTP/1.1 200 OK **Server: BIP Agent 4.15** **Content-Type: text/xml; charset="utf-8"** **Content-Length: 314** **<?xml version="1.0" encoding="UTF-8"?>** **<s:Envelope** **xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">** **<s:Body>** **<b:response xmlns:b="oid:/BIP">** **<b:slaveEndpointIRI>** **http://slave.example.org/BIP/kjfgsfa4146** **</b:slaveEndpointIRI>** **<b:masterEndpointIRI>** **http://master.example.org/BIP/fthjhuh5521** **</b:masterEndpointIRI>** **<b:linkNumber>197114367</b:linkNumber>** **<b:requestId>100000008</b:requestId>** **<b:deleteMaster/>** **<b:returnValue>0</b:returnValue>** **</b:response>** **</s:Body>** **</s:Envelope>**	9b Второстепенная конечная точка создает ответ **deleteMaster**. Связь ПМО БиоАПИ в этом случае будет уничтожена с помощью главной или второстепенной конечной точки

Приложение D

Разъяснение минимальных требований для простых систем

(обязательное)

D.1 Простая система с одним установленным биометрическим сканером

D.1.1 Данная система ПМО БиоАПИ обеспечивает обмены, основанные на модели системы ПМО БиоАПИ, в которой присутствует единственный ПБУ с единичным устройством получения данных, без съемный или заменяемых устройств или функциональных модулей. Данная система обеспечивает только базовые функции.

D.1.2 При применении этой системы будут осуществляться только начальные обмены, получение данных и загрузка ПБУ.

D.2 Простая система с одной базой данных ПБУ

D.2.1 Данная система ПМО БиоАПИ обеспечивает обмены, основанные на модели системы ПМО БиоАПИ, в которой присутствует единственное хранилище ПБУ, с возможностью создания и удаления базы данных.

D.2.2 При применении этой системы будут осуществляться только начальные обмены, имеется возможность использования открытой базы данных, и подмножества функций базы данных, исключая установки курсора и удаление ПБУ.

Приложение Е

Возможные сценарии, включающие в себя использование протокола межсетевого обмена БиоАПИ

(обязательное)

Важность использования стандартизированного протокола заключается в том, что он позволяет множеству поставщиков конкурировать за условие удаленного положения и условие центральной базы данных, и облегчает возможность модернизации будущих систем.

Е.1 Доступ в центральную государственную базу данных по управлению безопасности и здравоохранения

Е.1.1 За установку центральной государственной базы данных отвечает правительство. Обращение к даннной базе данных может быть обеспечено с помощью различных местных станций (пунктов въезда в страну, запросов об оказании медицинской помощи, пунктов доступа на засекреченные участки и т. д.).

Е.1.2 Доступ к базе данных основан на совокупности условий безопасности, связанных с получением доступа к системе (не относящейся к биометрии) и близостью человека, который может предоставить биометрические данные. Информация будет открыта только в том случае, если человек будет рядом со станцией (но, пребывая без сознания в чрезвычайных ситуациях или в ситуациях, опасных для жизни).

Е.1.3 Привилегии доступа (дополнительно к авторизации в местной станции) основаны на передаче биометрических данных с базы, расположенной на местной станции в центральный узел.

Е.1.4 ПМО БиоАПИ представляет собой стандартизированный, безопасный и гибкий способ поддержки такой деятельности.

Е.2 Регистрация людей в пункте входа или местном регистрационном центре

Е.2.1 Государство устанавливает центральную базу биометрических данных как основной вид идентификации людей. В каждом пункте входа биометрические данные, получаемые от человека, сверяют с базой данных, а записи соответственно обновляются со временем входа, выхода, и т.д.

Е.2.2 Если данные о человеке не присутствуют в базе данных, она обновляется, включая в себя идентификационные данные человека, а также его биометрические данные.

Е.2.3 Регистрационные центры также обеспечивают регистрацию людей в базе данных, с целью избежать задержек в пунктах выезда и въезда в страну.

Е.2.4 ПМО БиоАПИ представляет собой стандартизированный, безопасный и гибкий способ поддержки вышеуказанной деятельности.

Е.3 Доступ в места общественного досуга

Е.3.1 Место общественного досуга регистрирует в собственной базе данных биометрию всех клиентов с действительными билетами на пропускном пункте при их первичном посещении такого места.

Е.3.2 В таком случае посетители могут получить право входа или повторного входа (в пределах действия входного билета) в пропускном пункте любой зоны места общественного досуга, представляя свои биометрические данные.

Е.3.3 ПМО БиоАПИ представляет собой стандартизированный, безопасный и гибкий способ поддержки вышеуказанной деятельности.

Приложение F

Формальные модули ACH.1

(обязательное)

В настоящем приложении приведены все машинно-считываемые тексты содержимого спецификации ACH.1, а также необходимые заголовки модулей для обеспечения формального определения ACH.1.

```
-- BIP Module
        BIP {joint-iso-itu-t bip(41) modules(0) bip(0) version1(1)}
        DEFINITIONS AUTOMATIC TAGS ::=
        BEGIN

BIPMessage ::= SEQUENCE {
                nature                  CHOICE {
                        request                         BIPRequest,
                        response                BIPResponse,
                        notification            BIPNotification,
                        acknowledgement         BIPAcknowledgement
                        },
                ...
                }

BIPRequest ::= SEQUENCE {
                SlaveEndpointIRI                EndpointIRI,
                masterEndpointIRI               EndpointIRI,
                linkNumber                      UnsignedInt,
                requestId                       UnsignedInt,
                params                                  CHOICE {
                        addMaster                       AddMaster-RequestParams,
                        deleteMaster                    DeleteMaster-RequestParams,
                        bspLoad                         BSPLoad-RequestParams,
                        bspUnload                       BSPUnload-RequestParams,
                        queryUnits                      QueryUnits-RequestParams,
                        queryBFPs                       QueryBFPs-RequestParams,
                        bspAttach                       BSPAttach-RequestParams,
                        bspDetach                       BSPDetach-RequestParams,
                        enableUnitEvents                EnableUnitEvents-
                        RequestParams,
```

enableEventNotifications
EnableEventNotifications-RequestParams,
controlUnit ControlUnit-RequestParams,
control Control-RequestParams,
freeBIRHandle FreeBIRHandle-RequestParams,
getBIRFromHandle GetBIRFromHandle-
RequestParams,
getHeaderFromHandle
GetHeaderFromHandle-RequestParams,
subscribeToGUIEvents
SubscribeToGUIEvents-RequestParams,
unsubscribeFromGUIEvents
UnsubscribeFromGUIEvents-RequestParams,
redirectGUIEvents RedirectGUIEvents-
RequestParams,
unredirectGUIEvents
UnredirectGUIEvents-RequestParams,
queryGUIEventSubscriptions
QueryGUIEventSubscriptions-RequestParams,
notifyGUISelectEvent
NotifyGUISelectEvent-RequestParams,
notifyGUIStateEvent
NotifyGUIStateEvent-RequestParams,
notifyGUIProgressEvent
NotifyGUIProgressEvent-RequestParams,
capture Capture-RequestParams,
createTemplate CreateTemplate-RequestParams,
process Process-RequestParams,
processWithAuxBIR ProcessWithAuxBIR-
RequestParams,
verifyMatch VerifyMatch-RequestParams,
identifyMatch IdentifyMatch-RequestParams,
enroll Enroll-RequestParams,
verify Verify-RequestParams,
identify Identify-RequestParams,
import Import-RequestParams,
presetIdentifyPopulation
PresetIdentifyPopulation-RequestParams,
transform Transform-RequestParams,
dbOpen DbOpen-RequestParams,
dbClose DbClose-RequestParams,

```
                dbCreate                    DbCreate-RequestParams,
                dbDelete                    DbDelete-RequestParams,
                dbSetMarker                 DbSetMarker-RequestParams,
                dbFreeMarker                DbFreeMarker-RequestParams,
                dbStore                     DbStoreBIR-RequestParams,
                dbGetBIR                    DbGetBIR-RequestParams,
                dbGetNextBIR                DbGetNextBIR-RequestParams,
                dbDeleteBIR                 DbDeleteBIR-RequestParams,
                calibrateSensor             CalibrateSensor-RequestParams,
                setPowerMode                SetPowerMode-RequestParams,
                setIndicatorStatus
                                    SetIndicatorStatus-RequestParams,
                getIndicatorStatus
                                    GetIndicatorStatus-RequestParams,
                cancel                      Cancel-RequestParams,
                registerBSP                 RegisterBSP-RequestParams,
                unregisterBSP               UnregisterBSP-RequestParams,
                registerBFP                 RegisterBFP-RequestParams,
                unregisterBFP               UnregisterBFP-RequestParams,
        ...
        }
    }

BIPResponse ::= SEQUENCE {
        slaveEndpointIRI                    EndpointIRI,
        masterEndpointIRI                   EndpointIRI,
        linkNumber                          UnsignedInt,
        requestId                           UnsignedInt,
        params                              CHOICE {
            addMaster                       AddMaster-ResponseParams,
            deleteMaster                    DeleteMaster-ResponseParams,
            bspLoad                         BSPLoad-ResponseParams,
            bspUnload                       BSPUnload-ResponseParams,
            queryUnits                      QueryUnits-ResponseParams,
            queryBFPs                       QueryBFPs-ResponseParams,
            bspAttach                       BSPAttach-ResponseParams,
            bspDetach                       BSPDetach-ResponseParams,
            enableUnitEvents
                                    EnableUnitEvents-ResponseParams,
            enableEventNotifications
                                    EnableEventNotifications-ResponseParams,
```

controlUnit ControlUnit-ResponseParams,
control Control-ResponseParams,
freeBIRHandle FreeBIRHandle-ResponseParams,
getBIRFromHandle GetBIRFromHandle-ResponseParams,
getHeaderFromHandle GetHeaderFromHandle-ResponseParams,
subscribeToGUIEvents SubscribeToGUIEvents-ResponseParams,
unsubscribeFromGUIEvents UnsubscribeFromGUIEvents-ResponseParams,
redirectGUIEvents RedirectGUIEvents-ResponseParams,
unredirectGUIEvents UnredirectGUIEvents-ResponseParams,
queryGUIEventSubscriptions QueryGUIEventSubscriptions-ResponseParams,
notifyGUISelectEvent NotifyGUISelectEvent-ResponseParams,
notifyGUIStateEvent NotifyGUIStateEvent-ResponseParams,
notifyGUIProgressEvent NotifyGUIProgressEvent-ResponseParams,
capture Capture-ResponseParams,
createTemplate CreateTemplate-ResponseParams,
process Process-ResponseParams,
processWithAuxBIR ProcessWithAuxBIR-ResponseParams,
verifyMatch VerifyMatch-ResponseParams,
identifyMatch IdentifyMatch-ResponseParams,
enroll Enroll-ResponseParams,
verify Verify-ResponseParams,
identify Identify-ResponseParams,
import Import-ResponseParams,
presetIdentifyPopulation PresetIdentifyPopulation-ResponseParams,
transform Transform-ResponseParams,
dbOpen DbOpen-ResponseParams,
dbClose DbClose-ResponseParams,

```
                dbCreate                    DbCreate-ResponseParams,
                dbDelete                    DbDelete-ResponseParams,
                dbSetMarker                 DbSetMarker-ResponseParams,
                dbFreeMarker                DbFreeMarker-ResponseParams,
                dbStore                     DbStoreBIR-ResponseParams,
                dbGetBIR                    DbGetBIR-ResponseParams,
                dbGetNextBIR                DbGetNextBIR-ResponseParams,
                dbDeleteBIR                 DbDeleteBIR-ResponseParams,
                calibrateSensor             CalibrateSensor-
                ResponseParams,
                setPowerMode                SetPowerMode-ResponseParams,
                setIndicatorStatus
                                    SetIndicatorStatus-ResponseParams,
                getIndicatorStatus
                                    GetIndicatorStatus-ResponseParams,
                cancel                      Cancel-ResponseParams,
                registedBSP                 RegisterBSP-ResponseParams,
                unregisterBSP               UnregisterBSP-ResponseParams,
                registerBFP                 RegisterBFP-ResponseParams,
                unregisterBFP               UnregisterBFP-ResponseParams,
                ...
        },
        returnValue BioAPI-RETURN
    }
BIPNotification ::= SEQUENCE {
        masterEndpointIRI                   EndpointIRI,
        slaveEndpointIRI                    EndpointIRI,
        linkNumber                          UnsignedInt,
        notificationId                      UnsignedInt,
        params                              CHOICE {
            masterDeletionEvent
                                    MasterDeletionEvent-NotificationParams,
            unitEvent                       UnitEvent-NotificationParams,
            guiSelectEvent
                                    GUISelectEvent-NotificationParams,
            guiStateEvent
                                    GUIStateEvent-NotificationParams,
            guiProgressEvent
                                    GUIProgressEvent-NotificationParams,
            bspRegistrationEvent
                                    BSPRegistrationEvent-NotificationParams,
```

```
                bspUnregistrationEvent
                        BSPUnregistrationEvent-NotificationParams,
                bfpRegistrationEvent
                        BFPRegistrationEvent-NotificationParams,
                bfpUnregistrationEvent
                        BFPUnregistrationEvent-NotificationParams,
                ...
        }
    }

BIPAcknowledgement ::= SEQUENCE {
        masterEndpointIRI                   EndpointIRI,
        slaveEndpointIRI                    EndpointIRI,
        linkNumber                          UnsignedInt,
        notificationId                      UnsignedInt,
        params                              CHOICE {
            guiSelectEvent
                        GUISelectEvent-AcknowledgementParams,
            guiStateEvent
                        GUIStateEvent-AcknowledgementParams,
            guiProgressEvent
                        GUIProgressEvent-AcknowledgementParams,
            ...
        },
        returnValue BioAPI-RETURN
    }
    UnsignedByte        ::=     INTEGER (0..max-unsigned-byte)
    UnsignedShort       ::=     INTEGER (0..max-unsigned-short)
    UnsignedInt     ::= INTEGER (0..max-unsigned-int)
    SignedInt           ::=     INTEGER (min-signed-int..max-signed-int)
    MemoryAddress ::= INTEGER
    max-unsigned-byte   INTEGER ::= 255
    max-unsigned-short  INTEGER ::= 65535
    max-unsigned-int        INTEGER ::= 4294967295
    min-signed-int              INTEGER ::= -2147483648
    max-signed-int              INTEGER ::= 2147483647

EndpointIRI ::= VisibleString (CONSTRAINED BY
        {--The string shall conform to the "absolute-IRI" grammar--
        --defined in IETF RFC 3987--})
```

```
BioAPI-BFP-LIST-ELEMENT ::= SEQUENCE {
        category            BioAPI-CATEGORY,
        bfpProductUuid      BioAPI-UUID
    }

BioAPI-BFP-SCHEMA ::= SEQUENCE {
        bfpProductUuid      BioAPI-UUID,
        category            BioAPI-CATEGORY,
        description         BioAPI-STRING,
        path                UTF8String,
        specVersion         BioAPI-VERSION,
        productVersion      BioAPI-STRING,
        vendor              BioAPI-STRING,
        supportedFormats    SEQUENCE (SIZE(0..max-unsigned-int)) OF
                format BioAPI-BIR-BIOMETRIC-DATA-FORMAT,
        factorsMask         BioAPI-BIR-BIOMETRIC-TYPE,
        propertyUuid        BioAPI-UUID,
        property            BioAPI-DATA,
        hostingEndpointIRI  EndpointIRI
    }

BioAPI-BIR ::= SEQUENCE {
        patronFormatOwner   UnsignedShort,
        patronFormatType    UnsignedShort,
        formattedBIR        OCTET STRING
    }

BioAPI-BIR-ARRAY-POPULATION ::= SEQUENCE {
        members             SEQUENCE (SIZE(0..max-unsigned-int)) OF
        member              BioAPI-BIR
    }

BioAPI-BIR-BIOMETRIC-DATA-FORMAT ::= SEQUENCE {
        formatOwner         UnsignedShort,
        formatType          UnsignedShort
    }

BioAPI-BIR-BIOMETRIC-PRODUCT-ID ::= SEQUENCE {
        productOwner        UnsignedShort,
        productType         UnsignedShort
    }
```

```
BioAPI-BIR-BIOMETRIC-TYPE ::= BIT STRING {
        typeMultipleBiometricTypes  (0),
        typeFace                    (1),
        typeVoice                   (2),
        typeFinger                  (3),
        typeIris                    (4),
        typeRetina                  (5),
        typeHandGeometry            (6),
        typeSignatureSign           (7),
        typeKeystroke               (8),
        typeLipMovement             (9),
        typeGait                    (12),
        typeVein                    (13),
        typeDNA                     (14),
        typeEar                     (15),
        typeFoot                    (16),
        typeScent                   (17),
        typeOther                   (30),
        typePassword                (31)
    } (SIZE(32))

BioAPI-BIR-DATA-TYPE ::= SEQUENCE {
        processedLevel              ENUMERATED {
            raw,
            intermediate,
            processed,
            ...
        },
        flags                   BIT STRING {
            encrypted           (0),
            signed              (1),
            index-present       (3)
        } (SIZE(4))
    }

BioAPI-BIR-HANDLE ::= SignedInt

BioAPI-BIR-HEADER ::= SEQUENCE {
        patronFormatOwner   UnsignedShort,
        patronFormatType    UnsignedShort,
```

```
        formattedBIR            OCTET STRING
    }

BioAPI-BIR-PURPOSE ::= ENUMERATED {
        verify,
        identify,
        enroll,
        enrollVerify,
        enrollIdentify,
        audit,
        any,
        ...
    }

BioAPI-BIR-SECURITY-BLOCK-FORMAT ::= SEQUENCE {
        formatOwner             UnsignedShort,
        formatType              UnsignedShort
    }

BioAPI-BIR-SUBTYPE ::= CHOICE {
    anySubtype BIT STRING {
        left                    (0),
        right                   (1),
        thumb                   (2),
        pointerFinger           (3),
        middleFinger            (4),
        ringFinger              (5),
        littleFinger            (6)} (SIZE(7)),
    vein-only-subtype BIT STRING {
        left                    (0),
        right                   (1),
        veinPalm                (2),
        veinBackofhand          (3),
        veinWrist               (4)} (SIZE(7))
        }

BioAPI-BIR-SUBTYPE-MASK ::= BIT STRING {
        left                    (0),
        right                   (1),
        left-thumb              (2),
        left-pointerfinger      (3),
```

```
                left-middlefinger       (4),
                left-ringfinger         (5),
                left-littlefinger       (6),
                right-thumb             (7),
                right-pointerfinger     (8),
                right-middlefinger      (9),
                right-ringfinger        (10),
                right-littlefinger      (11),
                left-vein-palm          (12),
                left-vein-backofhand    (13),
                left-vein-wrist         (14),
                right-vein-palm         (15),
                right-vein-backofhand   (16),
                right-vein-wrist        (17)
        } (SIZE(32))

BioAPI-BSP-SCHEMA ::= SEQUENCE {
                bspProductUuid          BioAPI-UUID,
                description             BioAPI-STRING,
                path                    UTF8String,
                specVersion             BioAPI-VERSION,
                productVersion          BioAPI-STRING,
                vendor                  BioAPI-STRING,
                supportedFormats        SEQUENCE
                                (SIZE(0..max-unsigned-int)) OF
                                        format BioAPI-BIR-BIOMETRIC-DATA-FORMAT,
                factorsMask             BioAPI-BIR-BIOMETRIC-TYPE,
                operations              BioAPI-OPERATIONS-MASK,
                options                 BioAPI-OPTIONS-MASK,
                payloadPolicy           BioAPI-FMR,
                maxPayloadSize          UnsignedInt,
                defaultVerifyTimeout    SignedInt,
                defaultIdentifyTimeout  SignedInt,
                defaultCaptureTimeout   SignedInt,
                defaultEnrollTimeout    SignedInt,
                defaultCalibrateTimeout SignedInt,
                maxBSPDbSize            UnsignedInt,
                maxIdentify             UnsignedInt,
                hostingEndpointIRI      EndpointIRI,
                bspAccessUuid           BioAPI-UUID
        }
```

```
BioAPI-CANDIDATE ::= SEQUENCE {
        bir                         CHOICE {
                birInDatabase       BioAPI-UUID,
                birInArray          UnsignedInt,
                birInPresetArray    UnsignedInt
        },
        fmrAchieved                 BioAPI-FMR
}

BioAPI-CATEGORY ::= ENUMERATED {
        archive,
        comparisonAlgorithm,
        processingAlgorithm,
        sensor,
        ...
}

BioAPI-DATA ::= OCTET STRING (SIZE(0..max-unsigned-int))

BioAPI-DATE ::= SEQUENCE {
        year        INTEGER (0 | 1900..9999),
        month       INTEGER (0..12),
        day         INTEGER (0..31)
}

BioAPI-DB-ACCESS-TYPE ::= BIT STRING {
        read        (0),
        write       (1)
} (SIZE(32))

BioAPI-DB-MARKER-HANDLE ::= UnsignedInt

BioAPI-DB-HANDLE ::=    SignedInt

BioAPI-DBBIR-ID ::= SEQUENCE {
        dbHandle    BioAPI-DB-HANDLE,
        keyValue    BioAPI-UUID
}

BioAPI-DTG ::= SEQUENCE {
```

```
        date            BioAPI-DATE,
        time            BioAPI-TIME
    }

BioAPI-UNIT-EVENT-TYPE ::= ENUMERATED {
        insert,
        remove,
        fault,
        sourcePresent,
        sourceRemoved,
        ...
    }

BioAPI-UNIT-EVENT-TYPE-MASK ::= BIT STRING {
        insert              (0),
        remove              (1),
        fault               (2),
        sourcePresent       (3),
        sourceRemoved       (4)
    } (SIZE(32))

BioAPI-FMR ::= SignedInt

BioAPI-FRAMEWORK-SCHEMA ::= SEQUENCE {
        fwProductUuid       BioAPI-UUID,
        description         BioAPI-STRING,
        path UTF8String,
        specVersion         BioAPI-VERSION,
        productVersion      BioAPI-STRING,
        vendor              BioAPI-STRING,
        propertyUuid        BioAPI-UUID,
        property            BioAPI-DATA,
        hostingEndpointIRI  EndpointIRI
    }

BioAPI-GUI-BITMAP ::= SEQUENCE {
        subtypeMask         BioAPI-BIR-SUBTYPE-MASK,
        width               UnsignedInt,
        height              UnsignedInt,
        bitmap          BioAPI-DATA OPTIONAL
    }
```

```
BioAPI-GUI-BITMAP-ARRAY ::= SEQUENCE {
        guiBitmaps          SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                guiBitmap BioAPI-GUI-BITMAP
    }

BioAPI-GUI-EVENT-SUBSCRIPTION ::= SEQUENCE {
        subscriberEndpointIRI           EndpointIRI,
        guiEventSubscriptionUuid        BioAPI-UUID,
        guiSelectEventSubscribed        BOOLEAN,
        guiStateEventSubscribed         BOOLEAN,
        guiProgressEventSubscribed      BOOLEAN
    }

BioAPI-GUI-MOMENT ::= ENUMERATED {
        beforeStart,
        during,
        afterEnd,
        ...
    }

BioAPI-GUI-ENROLL-TYPE ::= BIT STRING {
        testVerify          (0),
        multipleCapture     (1)
    } (SIZE(32))

BioAPI-GUI-OPERATION ::= ENUMERATED {
        capture,
        process,
        createtemplate,
        verifymatch,
        identifymatch,
        verify,
        identify,
        enroll,
        ...
    }

BioAPI-GUI-RESPONSE ::= ENUMERATED {
        default,
        opComplete,
```

```
                opCancel,
                cycleStart,
                cycleRestart,
                subopStart,
                subopNext,
                progressContinue,
                progressCancel,
                recapture,
                ...
        }

BioAPI-GUI-SUBOPERATION ::= ENUMERATED {
                capture,
                process,
                createtemplate,
                verifymatch,
                identifymatch,
                ...
        }

    BioAPI-HANDLE ::= UnsignedInt

    BioAPI-IDENTIFY-POPULATION ::= SEQUENCE {
                birs                    CHOICE {
                        birDataBase     BioAPI-DB-HANDLE,
                        birArray        BioAPI-BIR-ARRAY-POPULATION,
                        birPresetArray  NULL
                }
        }

    BioAPI-INDICATOR-STATUS ::= ENUMERATED {
                accept,
                reject,
                ready,
                busy,
                failure,
                ...
        }

    BioAPI-INPUT-BIR ::= SEQUENCE {
                inputBIR                CHOICE {
```

```
                birInDB             BioAPI-DBBIR-ID,
                birInBSP            BioAPI-BIR-HANDLE,
                bir                 BioAPI-BIR
        }
    }

BioAPI-OPERATIONS-MASK ::= BIT STRING {
        enableEvents                (0),
        subscribeToGUIEvents        (1),
        capture                     (2),
        createTemplate              (3),
        process                     (4),
        processWithAuxBir           (5),
        verifyMatch                 (6),
        identifyMatch               (7),
        enroll                      (8),
        verify                      (9),
        identify                    (10),
        import                      (11),
        presetIdentifyPopulation    (12),
        databaseOperations          (13),
        setPowerMode                (14),
        setIndicatorStatus          (15),
        getIndicatorStatus          (16),
        calibrateSensor             (17),
        utilities                   (18),
        queryUnits                  (20),
        queryBFPs                   (21),
        controlUnit                 (22)
    } (SIZE(32))

BioAPI-OPTIONS-MASK ::= BIT STRING {
        raw                 (0),
        qualityRaw          (1),
        qualityIntermediate (2),
        qualityProcessed    (3),
        appGui              (4),
        guiProgressEvents   (5),
        sourcePresent       (6),
        payload             (7),
        birSign             (8),
```

```
                birEncrypt          (9),
                templateUpdate      (10),
                adaptation          (11),
                binning             (12),
                selfContainedDevice (13),
                moc                 (14),
                subtypeToCapture    (15),
                sensorBFP           (16),
                archiveBFP          (17),
                comparisonBFP       (18),
                processingBFP       (19),
                coarseScores        (20)
        } (SIZE(32))

BioAPI-POWER-MODE ::= ENUMERATED {
                normal,
                detect,
                sleep,
                ...
        }

BioAPI-QUALITY ::= INTEGER (-2..100)

BioAPI-RETURN ::= UnsignedInt

BioAPI-STRING ::= UTF8String (CONSTRAINED BY
                        {Закодированный UTF-8 не должен содержать параметров со
                        значением 0, и размер должен быть не более 268 октад })

BioAPI-TIME ::= SEQUENCE {
                hour        INTEGER (0..99),
                minute INTEGER (0..99),
                second      INTEGER (0..99)
        }

BioAPI-UNIT-ID ::= UnsignedInt

BioAPI-UNIT-LIST-ELEMENT ::= SEQUENCE {
                category    BioAPI-CATEGORY,
                unitID      BioAPI-UNIT-ID
        }
```

```
BioAPI-UNIT-SCHEMA ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID,
        unitManagerProductUuid  BioAPI-UUID,
        unitId                  BioAPI-UNIT-ID,
        category                BioAPI-CATEGORY,
        unitProperties          BioAPI-UUID,
        vendorInformation       BioAPI-STRING,
        supportedUnitEvents     BioAPI-UNIT-EVENT-TYPE-MASK,
        propertyUuid            BioAPI-UUID,
        property                BioAPI-DATA,
        hardwareVersion         BioAPI-STRING,
        firmwareVersion         BioAPI-STRING,
        softwareVersion         BioAPI-STRING,
        hardwareSerialNumber    BioAPI-STRING,
        authenticatedHardware   BOOLEAN,
        maxBSPDbSize            UnsignedInt,
        maxIdentify             UnsignedInt
    }

BioAPI-UUID ::= OCTET STRING (SIZE(16))

BioAPI-VERSION ::= SEQUENCE {
        major           INTEGER (0..15),
        minor           INTEGER (0..15)
    }

MasterDeletionEvent-NotificationParams ::= NULL

AddMaster-RequestParams ::= SEQUENCE {
        bipVersion              BioAPI-VERSION
    }

AddMaster-ResponseParams ::= SEQUENCE {
        fwSchema                BioAPI-FRAMEWORK-SCHEMA OPTIONAL,
        bspSchemas              SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                    bspSchema BioAPI-BSP-SCHEMA,
        bfpSchemas              SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                    bfpSchema BioAPI-BFP-SCHEMA
    }
```

```
LinkCallParams ::= SEQUENCE {
        slaveEndpointIRI        EndpointIRI
    }

DeleteMaster-RequestParams ::= NULL

DeleteMaster-ResponseParams ::= NULL

UnlinkCallParams ::= SEQUENCE {
        slaveEndpointIRI        EndpointIRI
    }

EnumFrameworksCallOutputParams ::= SEQUENCE OF BioAPI-FRAMEWORK-SCHEMA

EnumBSPsCallOutputParams ::= SEQUENCE OF BioAPI-BSP-SCHEMA

EnumBFPsCallOutputParams ::= SEQUENCE OF BioAPI-BFP-SCHEMA

BSPLoad-RequestParams ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID,
        unitEventSubscription   BOOLEAN
    }

BSPLoad-ResponseParams ::= NULL

BSPLoadCallParams ::= SEQUENCE {
        bspUuid                 BioAPI-UUID,
        unitEventHandlerAddress MemoryAddress,
        unitEventHandlerContext MemoryAddress
    }

BSPUnload-RequestParams ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID,
        unitEventSubscription   BOOLEAN
    }

BSPUnload-ResponseParams ::= NULL

BSPUnloadCallParams ::= SEQUENCE {
        bspUuid                 BioAPI-UUID,
        unitEventHandlerAddress MemoryAddress,
```

```
        unitEventHandlerContext     MemoryAddress
    }

QueryUnits-RequestParams ::= SEQUENCE {
        bspProductUuid              BioAPI-UUID
    }

QueryUnits-ResponseParams ::= SEQUENCE {
        unitSchemas                 SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                        unitSchema BioAPI-UNIT-SCHEMA
    }

QueryBFPs-RequestParams ::= SEQUENCE {
        bspProductUuid              BioAPI-UUID
    }

QueryBFPs-ResponseParams ::= SEQUENCE {
        bfps                        SEQUENCE
                                    (SIZE(0..max-unsigned-int)) OF
                                        bfp BioAPI-BFP-LIST-ELEMENT
    }

BSPAttach-RequestParams ::= SEQUENCE {
        bspProductUuid              BioAPI-UUID,
        version                         BioAPI-VERSION,
        units                       SEQUENCE
                                    (SIZE(0..max-unsigned-int)) OF
                                        unit BioAPI-UNIT-LIST-ELEMENT
    }

BSPAttach-ResponseParams ::= SEQUENCE {
        newOriginalBSPHandle        BioAPI-HANDLE
    }

BSPAttachCallOutputParams ::= SEQUENCE {
        newBSPHandle                BioAPI-HANDLE
    }

BSPDetach-RequestParams ::= SEQUENCE {
        originalBSPHandle           BioAPI-HANDLE
    }
```

```
BSPDetach-ResponseParams ::= NULL

EnableUnitEvents-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        unitEvents              BioAPI-UNIT-EVENT-TYPE-MASK
    }

EnableUnitEvents-ResponseParams ::= NULL

EnableEventNotifications-RequestParams ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID,
        unitEventTypes          BioAPI-UNIT-EVENT-TYPE-MASK
    }

EnableEventNotifications-ResponseParams ::= NULL

EnableCallParams ::= SEQUENCE {
        bspUuid                 BioAPI-UUID,
        unitEventTypes          BioAPI-UNIT-EVENT-TYPE-MASK
    }

ControlUnit-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        unitID                  BioAPI-UNIT-ID,
        controlCode             UnsignedInt,
        inputData               BioAPI-DATA
    }

ControlUnit-ResponseParams ::= SEQUENCE {
        outputData              BioAPI-DATA
    }

    Control-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        unitID                  BioAPI-UNIT-ID,
        controlCode             BioAPI-UUID,
        inputData               BioAPI-DATA
    }

    Control-ResponseParams ::= SEQUENCE {
```

```
        outputData                  BioAPI-DATA
    }

    FreeBIRHandle-RequestParams ::= SEQUENCE {
        originalBSPHandle           BioAPI-HANDLE,
        birHandle                   BioAPI-BIR-HANDLE
    }

    FreeBIRHandle-ResponseParams ::= NULL

    GetBIRFromHandle-RequestParams ::= SEQUENCE {
        originalBSPHandle           BioAPI-HANDLE,
        birHandle                   BioAPI-BIR-HANDLE
    }

    GetBIRFromHandle-ResponseParams ::= SEQUENCE {
        bir                         BioAPI-BIR
    }

        GetHeaderFromHandle-RequestParams ::= SEQUENCE {
        originalBSPHandle           BioAPI-HANDLE,
        birHandle                   BioAPI-BIR-HANDLE
    }

    GetHeaderFromHandle-ResponseParams ::= SEQUENCE {
        header                  BioAPI-BIR-HEADER
    }

    SubscribeToGUIEvents-RequestParams ::= SEQUENCE {
        guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
        bspProductUuid BioAPI-UUID          OPTIONAL,
        originalBSPHandle BioAPI-HANDLE  OPTIONAL,
        guiSelectEventSubscribed            BOOLEAN,
        guiStateEventSubscribed             BOOLEAN,
        guiProgressEventSubscribed          BOOLEAN
    }

SubscribeToGUIEvents-ResponseParams ::= NULL

SubscribeToGUIEventsCallParams ::= SEQUENCE {
        guiEventSubscriptionUuid            BioAPI-UUID OPTIONAL,
```

```
                bspUuid BioAPI-UUID                 OPTIONAL,
                bspHandle BioAPI-HANDLE             OPTIONAL,
                guiSelectEventHandlerAddress        MemoryAddress,
                guiSelectEventHandlerContext        MemoryAddress,
                guiStateEventHandlerAddress         MemoryAddress,
                guiStateEventHandlerContext         MemoryAddress,
                guiProgressEventHandlerAddress      MemoryAddress,
                guiProgressEventHandlerContext      MemoryAddress
        }

UnsubscribeFromGUIEvents-RequestParams ::= SEQUENCE {
                guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
                bspProductUuid              BioAPI-UUID OPTIONAL,
                originalBSPHandle           BioAPI-HANDLE OPTIONAL,
                guiSelectEventSubscribed    BOOLEAN,
                guiStateEventSubscribed     BOOLEAN,
                guiProgressEventSubscribed BOOLEAN
        }

UnsubscribeFromGUIEvents-ResponseParams ::= NULL

UnsubscribeFromGUIEventsCallParams ::= SEQUENCE {
                guiEventSubscriptionUuid            BioAPI-UUID OPTIONAL,
                bspUuid                             BioAPI-UUID OPTIONAL,
                bspHandle                           BioAPI-HANDLE OPTIONAL,
                guiSelectEventHandlerAddress        MemoryAddress,
                guiSelectEventHandlerContext        MemoryAddress,
                guiStateEventHandlerAddress         MemoryAddress,
                guiStateEventHandlerContext         MemoryAddress,
                guiProgressEventHandlerAddress      MemoryAddress,
                guiProgressEventHandlerContext      MemoryAddress
        }

QueryGUIEventSubscriptions-RequestParams ::= SEQUENCE {
                bspProductUuid              BioAPI-UUID
        }

QueryGUIEventSubscriptions-ResponseParams ::= SEQUENCE {
                guiEventSubscriptions               SEQUENCE (SIZE(0..max-unsigned-int))
                OF
```

```
                                        subscription BioAPI-GUI-EVENT-
                                    SUBSCRIPTION
    }

NotifyGUISelectEvent-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI       EndpointIRI,
        guiEventSubscriptionUuid    BioAPI-UUID,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        enrollType                  BioAPI-GUI-ENROLL-TYPE,
        operation                   BioAPI-GUI-OPERATION,
        moment                      BioAPI-GUI-MOMENT,
        resultCode                  BioAPI-RETURN,
        maxNumEnrollSamples         UnsignedInt,
        selectableInstances         BioAPI-BIR-SUBTYPE-MASK,
        capturedInstances           BioAPI-BIR-SUBTYPE-MASK,
        text UTF8String             OPTIONAL
    }

NotifyGUISelectEvent-ResponseParams ::= SEQUENCE {
        selectedInstances           BioAPI-BIR-SUBTYPE-MASK,
        response                    BioAPI-GUI-RESPONSE
    }

NotifyGUIStateEvent-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI       EndpointIRI,
        guiEventSubscriptionUuid    BioAPI-UUID,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        operation                   BioAPI-GUI-OPERATION,
        suboperation                BioAPI-GUI-SUBOPERATION,
        purpose                     BioAPI-BIR-PURPOSE,
        moment                      BioAPI-GUI-MOMENT,
        resultCode                  BioAPI-RETURN,
        enrollSampleIndex           SignedInt,
        bitmaps                     BioAPI-GUI-BITMAP-ARRAY
                                        OPTIONAL,
        text UTF8String                 OPTIONAL
    }

NotifyGUIStateEvent-ResponseParams ::= SEQUENCE {
```

```
        response                        BioAPI-GUI-RESPONSE,
        enrollSampleIndexToRecapture    SignedInt
    }

NotifyGUIProgressEvent-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI       EndpointIRI,
        guiEventSubscriptionUuid    BioAPI-UUID,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        operation                   BioAPI-GUI-OPERATION,
        suboperation                BioAPI-GUI-SUBOPERATION,
        purpose                     BioAPI-BIR-PURPOSE,
        moment                      BioAPI-GUI-MOMENT,
        suboperationProgress        UnsignedByte,
        bitmaps                     BioAPI-GUI-BITMAP-ARRAY
                                                OPTIONAL,
        text UTF8String             OPTIONAL
    }

NotifyGUIProgressEvent-ResponseParams ::= SEQUENCE {
        response                    BioAPI-GUI-RESPONSE
    }

RedirectGUIEvents-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI       EndpointIRI,
        guiEventSubscriptionUuid    BioAPI-UUID,
        originalBSPHandle           BioAPI-HANDLE,
        guiSelectEventRedirected    BOOLEAN,
        guiStateEventRedirected     BOOLEAN,
        guiProgressEventRedirected  BOOLEAN
    }

RedirectGUIEvents-ResponseParams ::= NULL

UnredirectGUIEvents-RequestParams ::= SEQUENCE {
        subscriberEndpointIRI       EndpointIRI,
        guiEventSubscriptionUuid    BioAPI-UUID,
        originalBSPHandle           BioAPI-HANDLE,
        guiSelectEventRedirected    BOOLEAN,
        guiStateEventRedirected     BOOLEAN,
        guiProgressEventRedirected  BOOLEAN
```

```
        }

UnredirectGUIEvents-ResponseParams ::= NULL

Capture-RequestParams ::= SEQUENCE {
                originalBSPHandle           BioAPI-HANDLE,
                purpose                     BioAPI-BIR-PURPOSE,
                subtype                     BioAPI-BIR-SUBTYPE,
                outputFormat                BioAPI-BIR-BIOMETRIC-DATA-FORMAT
                OPTIONAL,
                timeout                     SignedInt,
                no-auditData                BOOLEAN
        }

Capture-ResponseParams ::= SEQUENCE {
                capturedBIR                 BioAPI-BIR-HANDLE,
                auditData                   BioAPI-BIR-HANDLE OPTIONAL
        }

CreateTemplate-RequestParams ::= SEQUENCE {
                originalBSPHandle           BioAPI-HANDLE,
                capturedBIR                 BioAPI-INPUT-BIR,
                referenceTemplate           BioAPI-INPUT-BIR OPTIONAL,
                outputFormat                BioAPI-BIR-BIOMETRIC-DATA-FORMAT
                OPTIONAL,
                payload                     BioAPI-DATA OPTIONAL,
                no-templateUuid             BOOLEAN
        }

CreateTemplate-ResponseParams ::= SEQUENCE {
                newTemplate                 BioAPI-BIR-HANDLE,
                templateUuid                BioAPI-UUID OPTIONAL
        }

Process-RequestParams ::= SEQUENCE {
                originalBSPHandle           BioAPI-HANDLE,
                capturedBIR                 BioAPI-INPUT-BIR,
                outputFormat                BioAPI-BIR-BIOMETRIC-DATA-FORMAT
                OPTIONAL
        }
```

```
Process-ResponseParams ::= SEQUENCE {
        processedBIR            BioAPI-BIR-HANDLE
    }

ProcessWithAuxBIR-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        capturedBIR             BioAPI-INPUT-BIR,
        auxiliaryData           BioAPI-INPUT-BIR,
        outputFormat            BioAPI-BIR-BIOMETRIC-DATA-FORMAT
        OPTIONAL
    }

ProcessWithAuxBIR-ResponseParams ::= SEQUENCE {
        processedBIR            BioAPI-BIR-HANDLE
    }

VerifyMatch-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        maxFMRRequested         BioAPI-FMR,
        processedBIR            BioAPI-INPUT-BIR,
        referenceTemplate       BioAPI-INPUT-BIR,
        no-adaptedBIR           BOOLEAN,
        no-fmrAchieved          BOOLEAN,
        no-payload              BOOLEAN
    }

VerifyMatch-ResponseParams ::= SEQUENCE {
        adaptedBIR              BioAPI-BIR-HANDLE OPTIONAL,
        result                  BOOLEAN,
        fmrAchieved             BioAPI-FMR OPTIONAL,
        payload                 BioAPI-DATA OPTIONAL
    }

IdentifyMatch-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        maxFMRRequested         BioAPI-FMR,
        processedBIR            BioAPI-INPUT-BIR,
        population              BioAPI-IDENTIFY-POPULATION,
        totalNumberOfTemplates  UnsignedInt,
        binning                 BOOLEAN,
        maxNumberOfResults      UnsignedInt,
```

```
        timeout                 SignedInt
    }

IdentifyMatch-ResponseParams ::= SEQUENCE {
        candidates              SEQUENCE
                                    (SIZE(0..max-unsigned-int)) OF
                                        candidate BioAPI-CANDIDATE
    }

Enroll-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        purpose                 BioAPI-BIR-PURPOSE,
        subtype                 BioAPI-BIR-SUBTYPE,
        outputFormat            BioAPI-BIR-BIOMETRIC-DATA-FORMAT
        OPTIONAL,
        referenceTemplate       BioAPI-INPUT-BIR OPTIONAL,
        payload                 BioAPI-DATA OPTIONAL,
        timeout                 SignedInt,
        no-auditData            BOOLEAN,
        no-templateUuid         BOOLEAN
    }

Enroll-ResponseParams ::= SEQUENCE {
        newTemplate             BioAPI-BIR-HANDLE,
        auditData               BioAPI-BIR-HANDLE OPTIONAL,
        templateUuid            BioAPI-UUID OPTIONAL
    }

Verify-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        maxFMRRequested         BioAPI-FMR,
        referenceTemplate       BioAPI-INPUT-BIR,
        subtype                 BioAPI-BIR-SUBTYPE,
        timeout                 SignedInt,
        no-adaptedBIR           BOOLEAN,
        no-fmrAchieved          BOOLEAN,
        no-payload              BOOLEAN,
        no-auditData            BOOLEAN
    }

Verify-ResponseParams ::= SEQUENCE {
```

```
        adaptedBIR              BioAPI-BIR-HANDLE OPTIONAL,
        result                  BOOLEAN,
        fmrAchieved             BioAPI-FMR OPTIONAL,
        payload                 BioAPI-DATA OPTIONAL,
        auditData               BioAPI-BIR-HANDLE OPTIONAL
    }

Identify-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        maxFMRRequested         BioAPI-FMR,
        subtype                 BioAPI-BIR-SUBTYPE,
        population              BioAPI-IDENTIFY-POPULATION,
        totalNumberOfTemplates  UnsignedInt,
        binning                 BOOLEAN,
        maxNumberOfResults      UnsignedInt,
        timeout                 SignedInt,
        no-auditData            BOOLEAN
    }

Identify-ResponseParams ::= SEQUENCE {
        candidates              SEQUENCE
                                    (SIZE(0..max-unsigned-int)) OF
                                        candidate BioAPI-CANDIDATE,
        auditData               BioAPI-BIR-HANDLE OPTIONAL
    }

Import-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        inputData               BioAPI-DATA,
        inputFormat             BioAPI-BIR-BIOMETRIC-DATA-FORMAT,
        outputFormat            BioAPI-BIR-BIOMETRIC-DATA-FORMAT
        OPTIONAL,
        purpose                 BioAPI-BIR-PURPOSE
    }

Import-ResponseParams ::= SEQUENCE {
        constructedBIR          BioAPI-BIR-HANDLE
    }

PresetIdentifyPopulation-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
```

```
            population                  BioAPI-IDENTIFY-POPULATION
        }

PresetIdentifyPopulation-ResponseParams ::= NULL

Transform-RequestParams ::= SEQUENCE {
            bspHandle                   BioAPI-HANDLE,
            operationUuid               BioAPI-UUID,
            inputBIRs                   SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                            BioAPI-INPUT-BIR
        }

Transform-ResponseParams ::= SEQUENCE {
            outputBIRs                  SEQUENCE (SIZE(0..max-unsigned-int)) OF
                                            BioAPI-BIR-HANDLE
        }

DbOpen-RequestParams ::= SEQUENCE {
            originalBSPHandle           BioAPI-HANDLE,
            dbUuid                      BioAPI-UUID,
            accessRequest               BioAPI-DB-ACCESS-TYPE
        }

DbOpen-ResponseParams ::= SEQUENCE {
            dbHandle                    BioAPI-DB-HANDLE,
            markerHandle                BioAPI-DB-MARKER-HANDLE
        }

DbClose-RequestParams ::= SEQUENCE {
            originalBSPHandle           BioAPI-HANDLE,
            dbHandle                    BioAPI-DB-HANDLE
        }

DbClose-ResponseParams ::= NULL

DbCreate-RequestParams ::= SEQUENCE {
            originalBSPHandle           BioAPI-HANDLE,
            dbUuid                      BioAPI-UUID,
            numberOfRecords             UnsignedInt,
            accessRequest               BioAPI-DB-ACCESS-TYPE
        }
```

```
DbCreate-ResponseParams ::= SEQUENCE {
        dbHandle                BioAPI-DB-HANDLE
    }

DbDelete-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        dbUuid                  BioAPI-UUID
    }

DbDelete-ResponseParams ::= NULL

DbSetMarker-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        dbHandle                BioAPI-DB-HANDLE,
        keyValue                BioAPI-UUID,
        markerHandle            BioAPI-DB-MARKER-HANDLE
    }

DbSetMarker-ResponseParams ::= NULL

DbFreeMarker-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        markerHandle            BioAPI-DB-MARKER-HANDLE
    }

DbFreeMarker-ResponseParams ::= NULL

DbStoreBIR-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        birToStore              BioAPI-INPUT-BIR,
        dbHandle                BioAPI-DB-HANDLE
    }

DbStoreBIR-ResponseParams ::= SEQUENCE {
        birUuid                 BioAPI-UUID
    }

DbGetBIR-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        dbHandle                BioAPI-DB-HANDLE,
```

```
        keyValue                    BioAPI-UUID
    }

DbGetBIR-ResponseParams ::= SEQUENCE {
        retrievedBIR                BioAPI-BIR-HANDLE,
        markerHandle                BioAPI-DB-MARKER-HANDLE
    }

DbGetNextBIR-RequestParams ::= SEQUENCE {
        originalBSPHandle           BioAPI-HANDLE,
        dbHandle                    BioAPI-DB-HANDLE,
        markerHandle                BioAPI-DB-MARKER-HANDLE
    }

DbGetNextBIR-ResponseParams ::= SEQUENCE {
        retrievedBIR                BioAPI-BIR-HANDLE,
        birUuid                     BioAPI-UUID
    }

DbDeleteBIR-RequestParams ::= SEQUENCE {
        originalBSPHandle           BioAPI-HANDLE,
        dbHandle                    BioAPI-DB-HANDLE,
        keyValue                    BioAPI-UUID
    }

DbDeleteBIR-ResponseParams ::= NULL

CalibrateSensor-RequestParams ::= SEQUENCE {
        originalBSPHandle           BioAPI-HANDLE,
        timeout                     SignedInt
    }

CalibrateSensor-ResponseParams ::= NULL

SetPowerMode-RequestParams ::= SEQUENCE {
        originalBSPHandle           BioAPI-HANDLE,
        unitID                      BioAPI-UNIT-ID,
        powerMode                   BioAPI-POWER-MODE
    }

SetPowerMode-ResponseParams ::= NULL
```

```
SetIndicatorStatus-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        unitID                  BioAPI-UNIT-ID,
        indicatorStatus         BioAPI-INDICATOR-STATUS
    }

SetIndicatorStatus-ResponseParams ::= NULL

GetIndicatorStatus-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE,
        unitID                  BioAPI-UNIT-ID
    }

GetIndicatorStatus-ResponseParams ::= SEQUENCE {
        indicatorStatus         BioAPI-INDICATOR-STATUS
    }

Cancel-RequestParams ::= SEQUENCE {
        originalBSPHandle       BioAPI-HANDLE
    }

Cancel-ResponseParams ::= NULL

RegisterBSP-RequestParams ::= SEQUENCE {
        bspSchema               BioAPI-BSP-SCHEMA,
        update                  BOOLEAN
    }

RegisterBSP-ResponseParams ::= NULL

BSPRegistrationEvent-NotificationParams ::= SEQUENCE {
        bspSchema               BioAPI-BSP-SCHEMA,
        update                  BOOLEAN
    }

UnregisterBSP-RequestParams ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID
    }

UnregisterBSP-ResponseParams ::= NULL
```

```
BSPUnregistrationEvent-NotificationParams ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID
    }

RegisterBFP-RequestParams ::= SEQUENCE {
        bfpSchema               BioAPI-BFP-SCHEMA,
        update                  BOOLEAN
    }

RegisterBFP-ResponseParams ::= NULL

BFPRegistrationEvent-NotificationParams ::= SEQUENCE {
        bfpSchema               BioAPI-BFP-SCHEMA,
        update                  BOOLEAN
    }

    UnregisterBFP-RequestParams ::= SEQUENCE {
        bfpProductUuid          BioAPI-UUID
    }

UnregisterBFP-ResponseParams ::= NULL

BFPUnregistrationEvent-NotificationParams ::= SEQUENCE {
        bfpProductUuid          BioAPI-UUID
    }

UnitEvent-NotificationParams ::= SEQUENCE {
        bspProductUuid          BioAPI-UUID,
        unitID                  BioAPI-UNIT-ID,
        unitSchema              BioAPI-UNIT-SCHEMA OPTIONAL,
        unitEventType           BioAPI-UNIT-EVENT-TYPE
    }

UnitEventHandlerCallbackParams ::= SEQUENCE {
        unitEventHandlerAddress MemoryAddress,
        unitEventHandlerContext MemoryAddress,
        bspUuid                 BioAPI-UUID,
        unitID                  BioAPI-UNIT-ID,
        unitSchema              BioAPI-UNIT-SCHEMA OPTIONAL,
        unitEventType           BioAPI-UNIT-EVENT-TYPE
```

```
        }

    UnitEventInfo ::= SEQUENCE {
            hostingEndpointIRI          EndpointIRI,
            bspProductUuid              BioAPI-UUID,
            unitID                      BioAPI-UNIT-ID,
            unitSchema                  BioAPI-UNIT-SCHEMA OPTIONAL,
            unitEventType               BioAPI-UNIT-EVENT-TYPE
        }

    GUISelectEvent-NotificationParams ::= SEQUENCE {
            guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
            bspProductUuid              BioAPI-UUID,
            unitID                      BioAPI-UNIT-ID,
            originalBSPHandle           BioAPI-HANDLE OPTIONAL,
            enrollType                  BioAPI-GUI-ENROLL-TYPE,
            operation                   BioAPI-GUI-OPERATION,
            moment                      BioAPI-GUI-MOMENT,
            resultCode                  BioAPI-RETURN,
            maxNumEnrollSamples         UnsignedInt,
            selectableInstances         BioAPI-BIR-SUBTYPE-MASK,
            capturedInstances           BioAPI-BIR-SUBTYPE-MASK,
            text                        UTF8String OPTIONAL
        }

    GUISelectEvent-AcknowledgementParams ::= SEQUENCE {
            selectedInstances           BioAPI-BIR-SUBTYPE-MASK,
            response                    BioAPI-GUI-RESPONSE
        }

    GUISelectEventHandlerCallbackParams ::= SEQUENCE {
            guiSelectEventHandlerAddress    MemoryAddress,
            guiSelectEventHandlerContext    MemoryAddress,
            bspUuid                         BioAPI-UUID,
            unitID                          BioAPI-UNIT-ID,
            bspHandle                       BioAPI-HANDLE OPTIONAL,
            enrollType                      BioAPI-GUI-ENROLL-TYPE,
            operation                       BioAPI-GUI-OPERATION,
            moment                          BioAPI-GUI-MOMENT,
            resultCode                      BioAPI-RETURN,
            maxNumEnrollSamples             UnsignedInt,
```

```
        selectableInstances             BioAPI-BIR-SUBTYPE-MASK,
        capturedInstances               BioAPI-BIR-SUBTYPE-MASK,
        text                            UTF8String OPTIONAL
    }

GUISelectEventInfo ::= SEQUENCE {
        subscriberEndpointIRI           EndpointIRI,
        guiEventSubscriptionUuid        BioAPI-UUID OPTIONAL,
        hostingEndpointIRI              EndpointIRI,
        bspProductUuid                  BioAPI-UUID,
        unitID                          BioAPI-UNIT-ID,
        originalBSPHandle               BioAPI-HANDLE OPTIONAL,
        enrollType                      BioAPI-GUI-ENROLL-TYPE,
        operation                       BioAPI-GUI-OPERATION,
        moment                          BioAPI-GUI-MOMENT,
        resultCode                      BioAPI-RETURN,
        maxNumEnrollSamples             UnsignedInt,
        selectableInstances             BioAPI-BIR-SUBTYPE-MASK,
        capturedInstances               BioAPI-BIR-SUBTYPE-MASK,
        text                            UTF8String OPTIONAL
    }

GUIStateEvent-NotificationParams ::= SEQUENCE {
        guiEventSubscriptionUuid        BioAPI-UUID OPTIONAL,
        bspProductUuid                  BioAPI-UUID,
        unitID                          BioAPI-UNIT-ID,
        originalBSPHandle               BioAPI-HANDLE OPTIONAL,
        operation                       BioAPI-GUI-OPERATION,
        suboperation                    BioAPI-GUI-SUBOPERATION,
        purpose                         BioAPI-BIR-PURPOSE,
        moment                          BioAPI-GUI-MOMENT,
        resultCode                      BioAPI-RETURN,
        enrollSampleIndex               SignedInt,
        bitmaps                         BioAPI-GUI-BITMAP-ARRAY
                                            OPTIONAL,
        text                            UTF8String OPTIONAL
    }

GUIStateEvent-AcknowledgementParams ::= SEQUENCE {
        response                        BioAPI-GUI-RESPONSE,
        enrollSampleIndexToRecapture    SignedInt
```

```
}

GUIStateEventHandlerCallbackParams ::= SEQUENCE {
        guiStateEventHandlerAddress     MemoryAddress,
        guiStateEventHandlerContext     MemoryAddress,
        bspUuid                         BioAPI-UUID,
        unitID                          BioAPI-UNIT-ID,
        bspHandle                       BioAPI-HANDLE OPTIONAL,
        operation                       BioAPI-GUI-OPERATION,
        suboperation                    BioAPI-GUI-SUBOPERATION,
        purpose                         BioAPI-BIR-PURPOSE,
        moment                          BioAPI-GUI-MOMENT,
        resultCode                      BioAPI-RETURN,
        enrollSampleIndex               SignedInt,
        bitmaps                         BioAPI-GUI-BITMAP-ARRAY
                                        OPTIONAL,
        text                            UTF8String OPTIONAL
}

GUIStateEventInfo ::= SEQUENCE {
        subscriberEndpointIRI       EndpointIRI,
        guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
        hostingEndpointIRI          EndpointIRI,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        originalBSPHandle           BioAPI-HANDLE OPTIONAL,
        operation                   BioAPI-GUI-OPERATION,
        suboperation                BioAPI-GUI-SUBOPERATION,
        purpose                     BioAPI-BIR-PURPOSE,
        moment                      BioAPI-GUI-MOMENT,
        resultCode                  BioAPI-RETURN,
        enrollSampleIndex           SignedInt,
        bitmaps                     BioAPI-GUI-BITMAP-ARRAY
                                        OPTIONAL,
        text                        UTF8String OPTIONAL
}

GUIProgressEvent-NotificationParams ::= SEQUENCE {
        guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
```

```
        originalBSPHandle           BioAPI-HANDLE OPTIONAL,
        operation                   BioAPI-GUI-OPERATION,
        suboperation                BioAPI-GUI-SUBOPERATION,
        purpose                     BioAPI-BIR-PURPOSE,
        moment                      BioAPI-GUI-MOMENT,
        suboperationProgress        UnsignedByte,
        bitmaps                     BioAPI-GUI-BITMAP-ARRAY
                                        OPTIONAL,
        text                        UTF8String OPTIONAL
    }

GUIProgressEvent-AcknowledgementParams ::= SEQUENCE {
        response BioAPI-GUI-RESPONSE
    }

GUIProgressEventHandlerCallbackParams ::= SEQUENCE {
        guiProgressEventHandlerAddress   MemoryAddress,
        guiProgressEventHandlerContext   MemoryAddress,
        bspUuid                          BioAPI-UUID,
        unitID                           BioAPI-UNIT-ID,
        bspHandle                        BioAPI-HANDLE OPTIONAL,
        operation                        BioAPI-GUI-OPERATION,
        suboperation                     BioAPI-GUI-SUBOPERATION,
        purpose                          BioAPI-BIR-PURPOSE,
        moment                           BioAPI-GUI-MOMENT,
        suboperationProgress             UnsignedByte,
        bitmaps                          BioAPI-GUI-BITMAP-ARRAY
                                             OPTIONAL,
        text                             UTF8String OPTIONAL
    }

GUIProgressEventInfo ::= SEQUENCE {
        subscriberEndpointIRI       EndpointIRI,
        guiEventSubscriptionUuid    BioAPI-UUID OPTIONAL,
        hostingEndpointIRI          EndpointIRI,
        bspProductUuid              BioAPI-UUID,
        unitID                      BioAPI-UNIT-ID,
        originalBSPHandle           BioAPI-HANDLE OPTIONAL,
        operation                   BioAPI-GUI-OPERATION,
        suboperation                BioAPI-GUI-SUBOPERATION,
        purpose                     BioAPI-BIR-PURPOSE,
```

```
            moment                  BioAPI-GUI-MOMENT,
            suboperationProgress    UnsignedByte,
            bitmaps                 BioAPI-GUI-BITMAP-ARRAY
                                        OPTIONAL,
            text                    UTF8String OPTIONAL
        }

VisibleEndpoints ::= SET OF endpoint VisibleEndpoint

VisibleEndpoint ::= BioAPI-FRAMEWORK-SCHEMA

VisibleBSPRegistrations ::= SET OF
            registration VisibleBSPRegistration

VisibleBSPRegistration ::= BioAPI-BSP-SCHEMA

VisibleBFPRegistrations ::= SET OF
            registration VisibleBFPRegistration

VisibleBFPRegistration ::= BioAPI-BFP-SCHEMA

RunningBSPLocalReferences ::= SET OF
            reference RunningBSPLocalReference

RunningBSPLocalReference ::= SEQUENCE {
            hostingEndpointIRI      EndpointIRI,
            bspProductUuid          BioAPI-UUID,
            useBSPAccessUuid        BOOLEAN,
            unitEventHandlerAddress MemoryAddress,
            unitEventHandlerContext MemoryAddress
        }

RunningBSPRemoteReferences ::= SET OF
            reference RunningBSPRemoteReference

RunningBSPRemoteReference ::= SEQUENCE {
            referrerEndpointIRI     EndpointIRI,
            bspProductUuid          BioAPI-UUID,
            unitEventSubscription   BOOLEAN
        }
```

```
UnitEventNotificationDisablers ::= SET OF
        disabler UnitEventNotificationDisabler

UnitEventNotificationDisabler ::= SEQUENCE {
        referrerEndpointIRI     EndpointIRI,
        bspProductUuid          BioAPI-UUID,
        unitEventTypes          BioAPI-UNIT-EVENT-TYPE-MASK
    }

AttachSessionLocalReferences ::= SET OF
        reference AttachSessionLocalReference

AttachSessionLocalReference ::= SEQUENCE {
        hostingEndpointIRI      EndpointIRI,
        bspProductUuid          BioAPI-UUID,
        useBSPAccessUuid        BOOLEAN,
        originalBSPHandle       BioAPI-HANDLE,
        localBSPHandle          BioAPI-HANDLE
    }

AttachSessionRemoteReferences ::= SET OF
        reference AttachSessionRemoteReference

AttachSessionRemoteReference ::= SEQUENCE {
        referrerEndpointIRI     EndpointIRI,
        bspProductUuid          BioAPI-UUID,
        originalBSPHandle       BioAPI-HANDLE
    }

GUIEventLocalSubscriptions ::= SET OF
        subscription GUIEventLocalSubscription

GUIEventLocalSubscription ::= SEQUENCE {
        guiEventSubscriptionUuid        BioAPI-UUID OPTIONAL,
        hostingEndpointIRI              EndpointIRI,
        bspProductUuid                  BioAPI-UUID,
        useBSPAccessUuid                BOOLEAN,
        originalBSPHandle               BioAPI-HANDLE OPTIONAL,
        guiSelectEventHandlerAddress    MemoryAddress,
        guiSelectEventHandlerContext    MemoryAddress,
        guiStateEventHandlerAddress     MemoryAddress,
```

```
        guiStateEventHandlerContext          MemoryAddress,
        guiProgressEventHandlerAddress       MemoryAddress,
        guiProgressEventHandlerContext       MemoryAddress
    }

GUIEventRemoteSubscriptions ::= SET OF
        subscription GUIEventRemoteSubscription

GUIEventRemoteSubscription ::= SEQUENCE {
        subscriberEndpointIRI                EndpointIRI,
        guiEventSubscriptionUuid             BioAPI-UUID OPTIONAL,
        bspProductUuid                       BioAPI-UUID,
        originalBSPHandle                    BioAPI-HANDLE OPTIONAL,
        guiSelectEventSubscribed             BOOLEAN,
        guiStateEventSubscribed              BOOLEAN,
        guiProgressEventSubscribed           BOOLEAN
    }

GUIEventRedirectors ::= SET OF
        redirector GUIEventRedirector

GUIEventRedirector ::= SEQUENCE {
        referrerEndpointIRI                  EndpointIRI,
        bspProductUuid                       BioAPI-UUID,
        originalBSPHandle                    BioAPI-HANDLE,
        subscriberEndpointIRI                EndpointIRI,
        guiEventSubscriptionUuid             BioAPI-UUID,
        guiSelectEventRedirected             BOOLEAN,
        guiStateEventRedirected              BOOLEAN,
        guiProgressEventRedirected           BOOLEAN
    }

ApplicationOwnedMemoryBlocks ::= SET OF
        memoryBlock ApplicationOwnedMemoryBlock

ApplicationOwnedMemoryBlock ::= SEQUENCE {
        address         MemoryAddress
    }

END
```

```
-- BIP-TCPIP Module

BIP-TCPIP {joint-iso-itu-t bip(41) modules(0) bip-tcpip(1) version1(1)}
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

        IMPORTS BIPMessage FROM BIP{joint-iso-itu-t bip(41) modules(0) bip(0) version1(1)};

TCPIPBIPMessage ::= SEQUENCE {
        magicNumber             OCTET STRING(SIZE(4))('3AC49E70'H),
        version                         INTEGER{version-1(1)}(0..255),
        content                         CHOICE {
                bIPMessage      OCTET STRING(CONTAINING BIPMessage
                                ENCODED BY basic-per-aligned),
                keepalive       NULL,
                requestLinkChannelOnSpecifiedPort INTEGER(0..65535),
                requestLinkChannel      NULL
                }
        }

basic-per-aligned OBJECT IDENTIFIER ::=
        {joint-iso-itu-t asn1(1) packed-encoding(3) basic(0) aligned(0)}

END

-- BIP-DISCOVERY Module

BIP-DISCOVERY {joint-iso-itu-t bip(41) modules(0) bip-discovery(2) version1(1)}
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

Discovery ::= SEQUENCE {
        protocolVersion                 ProtocolVersion,
        masterEndpointAddress           IPAddress,
        masterEndPort                   Port DEFAULT 4376,
        ...
        }

Announcement ::= SEQUENCE {
        protocolVersion                 ProtocolVersion,
        slaveEndpointIP                 Address IPAddress,
```

```
        slaveEndpointMACAddress   MACAddress,
        slaveEndpointName   IA5String(SIZE(1..32)),
        bipMessagePort      Port DEFAULT 4376,
        securityProtocols   SEQUENCE OF SecurityProtocol OPTIONAL,
        ...
        }

ProtocolVersion ::= SEQUENCE {
        major INTEGER(0..255),
        minor INTEGER(0..255)
        }

IPAddress ::= CHOICE {
        ipv4 OCTET STRING(SIZE(4)),
        ipv6 OCTET STRING(SIZE(16))
        }

Port ::= INTEGER(0..65535)

MACAddress ::= OCTET STRING(SIZE(6))

SecurityProtocol ::=SEQUENCE {
        id SECURITY-PROTOCOL.&id({SecurityProtocols}),
        parameter SECURITY-PROTOCOL.&Parameter({SecurityProtocols}{@id})
        }

SECURITY-PROTOCOL ::= CLASS {
        &id OBJECT IDENTIFIER,
        &Parameter
        }

SecurityProtocols SECURITY-PROTOCOL ::= {...}
END
```

Приложение G

(справочное)

Библиография

[1] W3C SOAP 1.1:2000, Simple Object Access Protocol

Приложение ДА

(справочное)

Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации

Сведения о соответствии ссылочных международных стандартов национальным стандартам Российской Федерации приведены в таблице ДА.1.

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО/МЭК 9834-8:2005	IDT	ГОСТ Р ИСО/МЭК 9834-8-2011 «Информационная технология. Взаимосвязь открытых систем. Процедуры работы уполномоченных по регистрации ВОС. Часть 8. Создание, регистрация универсально уникальных идентификаторов (УУИд) и их использование в качестве компонентов идентификатора объекта АСН.1»
ИСО/МЭК 8824-1:2002	-	*
ИСО/МЭК 8824-2:2002	-	*
ИСО/МЭК 8824-3:2002	-	*
ИСО/МЭК 8824-4:2002	-	*
ИСО/МЭК 8825-2:2002	-	*
ИСО/МЭК 8825-4:2002	IDT	ГОСТ Р ИСО/МЭК 8825-4-2009 «Информационная технология. Правила кодирования АСН.1. Часть 4. Правила XML кодирования (XER)»
ИСО/МЭК 8825-4:2002 /Доп.1:2004		

Продолжение таблицы ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО/МЭК ТО 8802-1:2001	-	*
ИСО/МЭК 19784-1:2006	IDT	ГОСТ Р ИСО/МЭК 19784-1–2007 «Автоматическая идентификация. Идентификация биометрическая. Биометрический программный интерфейс. Часть 1. Спецификация биометрического программного интерфейса»
ИСО/МЭК 19785-1:2006	IDT	ГОСТ Р ИСО/МЭК 19785-1–2008 «Автоматическая идентификация. Идентификация биометрическая. Единая структура форматов обмена биометрическими данными. Часть 1. Спецификация элементов данных»
ИСО/МЭК 19785-3:2007	-	*
ИСО/МЭК 19794 (все части)	IDT	Комплекс стандартов ГОСТ Р ИСО/МЭК 19794
IETF RFC 768 (1980)	-	*
IETF RFC 791 (1981)	-	*
IETF RFC 793 (1981)	-	*
IETF RFC 826 (1982)	-	*
IETF RFC 1945 (1996)	-	*
IETF RFC 2131 (1997)	-	*
IETF RFC 2136 (1997),	-	*
IETF RFC 2462 (1998),	-	*

Окончание таблицы ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
IETF RFC 2616 (1999)	-	*
IETF RFC 2818 (2000),	-	*
IETF RFC 3315 (2003),	-	*
IETF RFC 3987 (2005),	-	*
IETF RFC 4443 (2006),	-	*
W3C SOAP 1.2:2007	-	*
W3C SOAP MTOM:2005	-	*
W3C XMLENC:2002	-	*
W3C XMLDSIG:2002	-	*

*Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Примечание - В настоящей таблице использовано следующее условное обозначение степени соответствия стандарта:

- IDT – идентичный стандарт.

УДК004.93'1:006.89 ОКС35.040 П85

Ключевые слова: информационные технологии, биометрия, БиоАПИ, протокол межсетевого обмена

Подписано в печать 02.03.2015. Формат 60x84 1/8.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

ФГУП «СТАНДАРТИНФОРМ»,

123995 Москва, Гранатный пер., 4.

www.gostinfo.ru info@gostinfo.ru