ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО/МЭК 15961-1— 2023

Информационные технологии

ПРОТОКОЛ ДАННЫХ РАДИОЧАСТОТНОЙ ИДЕНТИФИКАЦИИ ДЛЯ УПРАВЛЕНИЯ ПРЕДМЕТАМИ

Часть 1

Прикладной интерфейс

(ISO/IEC 15961-1:2021, Information technology — Data protocol for radio frequency identification (RFID) for item management — Part 1: Application interface, IDT)

Издание официальное

Москва
Российский институт стандартизации
2023

gost.e-ecolog.ru

Предисловие

1 ПОДГОТОВЛЕН Ассоциацией автоматической идентификации «ЮНИСКАН/ГС1 РУС» (ГС1 РУС) совместно с Акционерным обществом «Дизайн центр «Кристал» (АО «ДЦ «Кристал») на основе собственного перевода на русский язык англоязычной версии стандарта, указанного в пункте 4

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

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 15 ноября 2023 г. № 1393-ст

4 Настоящий стандарт идентичен международному стандарту ИСО/МЭК 15961-1:2021 «Информационные технологии. Протокол данных для радиочастотной идентификации (РЧИ) для управления предметами. Часть 1. Прикладной интерфейс» (ISO/IEC 15961-1:2021 «Information technology — Data protocol for radio frequency identification (RFID) for item management — Part 1: Application interface», IDT).

Наименование настоящего стандарта изменено относительно наименования указанного международного стандарта для приведения в соответствие с ГОСТ Р 1.5—2012 (пункт 3.5).

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

Дополнительные сноски в тексте стандарта, выделенные курсивом, приведены для пояснения текста оригинала

5 ВВЕДЕН ВПЕРВЫЕ

6 Некоторые элементы настоящего стандарта могут быть объектами патентных прав. Международная организация по стандартизации (ИСО) и Международная электротехническая комиссия (МЭК) не несут ответственности за установление подлинности каких-либо или всех таких патентных прав

*Правила применения настоящего стандарта установлены в статье 26 Федерального закона от 29 июня 2015 г. № 162-ФЗ «О стандартизации в Российской Федерации». Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© ISO, 2021
© IEC, 2021
© Оформление. ФГБУ «Институт стандартизации», 2023

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

Содержание

1 Область применения 1
2 Нормативные ссылки 1
3 Термины, определения и сокращения 2
3.1 Термины и определения 2
3.2 Сокращения и обозначения 2
4 Соответствие стандартам 2
4.1 Общие положения 2
4.2 Соответствие требованиям 2
4.3 Соответствие процессора данных (Data Processor) 3
5 Модель протокола 3
6 Соглашения о представлении 3
6.1 Представление команд, ответов и аргументов 3
6.2 Представление идентификатора объекта в интерфейсе приложения 4
6.3 Байтовая нотация 5
7 Обработка команд и ответов приложения 6
7.1 Общие данные 6
7.2 Информация, связанная с системой кодирования в командах 7
7.3 Подготовка основных объектов и других аргументов основных приложений 10
7.4 Другие аргументы команды 14
7.5 Имена полей, связанных с командами 25
7.6 Обеспечение безопасности данных (Data security) 25
8 Потоки данных и процессы для радиоинтерфейса 26
8.1 Общие положения 26
8.2 Установление связи между приложением и радиочастотной меткой 26
8.3 Службы прикладной системы 26
9 Коды команд, коды завершения и коды выполнения 27
9.1 Общие положения 27
9.2 Значения конечных дуг модулей команд и ответов 27
9.3 Код завершения (Completion-Code) 28
9.4 Код выполнения (Execution-Code) 28
10 Команды и ответы на команды 29
10.1 Общие положения 29
10.2 Сконфигурировать идентификатор AFI (Configure-AFI) 29
10.3 Сконфигурировать идентификатор DSFID (Configure-DSFID) 30
10.4 Инвентаризировать радиочастотные метки (Inventory-Tags) 31
10.5 Удалить объект (Delete-Object) 33
10.6 Изменить объект (Modify-Object) 34
10.7 Считать идентификаторы объектов (Read-Object-Identifiers) 36
10.8 Считать карту логической памяти (Read-Logical-Memory-Map) 37
10.9 Очистить память (Erase-Memory) 38
10.10 Получить системную информацию, связанную с приложением (Get-App-Based-System-Info) 38
10.11 Записать объекты (Write-Objects) 39
10.12 Считать объекты (Read-Objects) 42
10.13 Записать объекты в сегментированную память радиочастотной метки (Write-Objects-Segmented-Memory-Tag) 43
10.14 Записать EPC-UII (Write-EPC-UII) 46
10.15 Инвентаризировать память ISO-UII (Inventory-ISO-UIImemory) 47
10.16 Инвентаризировать память EPC-UII (Inventory-EPC-UIImemory) 48
10.17 Записать пароль в сегментированную память радиочастотной метки (Write-Password-Segmented-Memory-Tag) 49
10.18 Считать слова из сегментированной памяти радиочастотной метки (Read-Words-Segmented-Memory-Tag) 50
10.19 Уничтожить сегментированную память радиочастотной метки (Kill-Segmented-Memory-Tag) 51

10.20 Удалить упакованный объект (Delete-Packed-Object) 51
10.21 Изменить структуру упакованного объекта (Modify-Packed-Object-Structure) 52
10.22 Записать сегменты радиочастотной метки типа 6TypeD (Write-Segments-6TypeD-Tag) 54
10.23 Считать сегменты радиочастотной метки типа 6TypeD (Read-Segments-6TypeD-Tag) 56
10.24 Записать мономорфный идентификатор UII (Write-Monomorphic-UII) 58
10.25 Сконфигурировать расширенный идентификатор DSFID (Configure-Extended-DSFID) 61
10.26 Сконфигурировать заголовок блочных записей (Configure-Multiple-Records-Header) 63
10.27 Считать блочные записи (Read-Multiple-Records) 65
10.28 Удалить блочную запись (Delete-Multiple-Record) 67
11 Аргументы 68
11.1 Аргумент Add-Objects (добавление объектов) 68
11.2 Аргумент DSFID-Constructs (конструкции идентификатора DSFID) 69
11.3 Аргумент EPC-UII-Memory (память EPC-UII) 70
11.4 Аргумент Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) 70
11.5 Аргумент ISO-UIImemory (память ISO-UII) 71
11.6 Аргумент Item-Related-Add-Objects (добавление объектов, связанных с предметами) 71
11.7 Аргумент Item-Related-DSFID-Constructs (конструкции идентификатора DSFID, связанного с предметом) 71
11.8 Аргумент Multiple-Records-Constructs (конструкции блочных записей) 71
11.9 Аргумент Multiple-Records-Directory-Structure (структура каталога блочных записей) 73
11.10 Аргумент Multiple-Records-Header-Structure (структура заголовка блочных записей) 74
11.11 Аргумент Multiple-Records-Preamble-Structure (структура преамбулы блочных записей) 75
11.12 Аргумент Packed-Objects-Constructs (конструкции упакованных объектов) 75
11.13 Аргумент Read-Objects (считывание объектов) 77
11.14 Ответ на считывание объектов (Read-Objects-Response) 77
11.15 Аргумент Read-OIDs-Response (ответ на считывание идентификаторов объектов) 77
11.16 Аргумент UII-Add-Objects (добавление объектов в формате идентификатора UII) 77
11.17 Аргумент UII-DSFID-Constructs (конструкции UII-DSFID) 78
11.18 Аргумент Write-Responses (ответы на команду записи) 78
Приложение А (рекомендуемое) Абстрактный синтаксис и правила кодирования передачи по ИСО/МЭК 15961:2004 79
Приложение В (рекомендуемое) Адаптация установленных форматов данных 86
Приложение С (рекомендуемое) Связанные объекты данных 87
Приложение D (рекомендуемое) Вопросы обеспечения безопасности данных 88
Приложение Е (рекомендуемое) Исходные команды и ответы с использованием абстрактного синтаксиса языка АСН.1 90
Приложение F (справочное) Пример кодирования передачи по ИСО/МЭК 15961:2004 116
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов национальным и межгосударственным стандартам 119
Библиография 120

Введение

Технология радиочастотной идентификации (РЧИ, radio frequency identification, RFID) основана на бесконтактной электронной связи через радиоинтерфейс. Структура битов, хранящихся в памяти радиочастотной метки, невидима и доступна только при использовании соответствующего протокола радиоинтерфейса между радиочастотной меткой и устройством опроса согласно соответствующей части серии ИСО/МЭК 18000*. Передача данных между приложением и устройством опроса в открытых системах требует, чтобы данные были представлены единообразно для любых радиочастотных меток, являющихся частью этой открытой системы. Команды, поступающие из приложения, и ответы от устройства опроса также требуют стандартной обработки. Это необходимо не только для обеспечения возможности взаимодействия оборудования, но и для использования особого носителя данных, чтобы данные, которые будут закодированы в радиочастотной метке в одной реализации системы, могли быть прочитаны позднее в совершенно другой и неизвестной реализации системы. Биты данных, хранящиеся в каждой радиочастотной метке, должны быть отформатированы таким образом, чтобы их можно было надежно прочитать в месте использования для выполнения радиочастотной меткой своей основной задачи. Эта надежность достигается с помощью спецификации протокола данных, указанного в настоящем стандарте, и правил кодирования данных по ИСО/МЭК 15962. Дополнительно ИСО/МЭК 24791-1 определяет систему программного обеспечения, которая поддерживает работу радиочастотной системы между прикладной системой для предприятий и устройством опроса. Конкретные компоненты стандартов инфраструктуры отвечают требованиям управления данными (ИСО/МЭК 24791-2) и требованиям устройства интерфейса (ИСО/МЭК 24791-5). Они поддерживают определенную реализацию, которая включает правила кодирования по ИСО/МЭК 15962 и функциональные правила команд и ответов в настоящем стандарте.

Изготовителям радиочастотного оборудования (устройств опроса, радиочастотных меток и т. д.) и пользователям радиочастотных технологий требуются протоколы данных на основе радиочастотных стандартов для управления предметами. Настоящий стандарт, ИСО/МЭК 15962, ИСО/МЭК 24791-1, ИСО/МЭК 24791-2 и ИСО/МЭК 24791-5 определяют такие протоколы, накладываемые на радиоинтерфейс, определенный в серии стандартов ИСО/МЭК 18000*.

Передача данных в устройство и из него, поддерживаемая соответствующими командами устройства, является предметом рассмотрения настоящего стандарта. Данный стандарт предназначен для использования в качестве руководящего материала при разработке программного обеспечения, соответствующего приложениям и оборудованию радиочастотной идентификации. ИСО/МЭК 15962, который предназначен для использования совместно с данным стандартом, определяет общий процесс и методологию, разработанные для форматирования данных приложения в структуру для хранения в радиочастотной метке.

Примечание — ИСО/МЭК 15961:2004 является отмененным стандартом, замененным на ИСО/МЭК 15961-1, ИСО/МЭК 15961-2, ИСО/МЭК 15961-3 и ИСО/МЭК 15961-4. Ссылки на ИСО/МЭК 15961:2004 указывают на различия с этим стандартом, поскольку некоторые системы по-прежнему используют отмененную редакцию стандарта. Вся информация, касающаяся использования отмененного стандарта ИСО/МЭК 15961:2004, содержится в настоящем стандарте. Цель этого состоит в том, чтобы удалить ссылку на отмененный стандарт в следующей редакции ИСО/МЭК 15961-1.

* *См.* [*10*] — [*18*].

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационные технологии

ПРОТОКОЛ ДАННЫХ РАДИОЧАСТОТНОЙ ИДЕНТИФИКАЦИИ ДЛЯ УПРАВЛЕНИЯ ПРЕДМЕТАМИ

Часть 1

Прикладной интерфейс

Information technology. Data protocol for radio frequency identification for item management. Part 1. Application interface

Дата введения — 2024—06—01

1 Область применения

Настоящий стандарт распространяется на абстрактный интерфейс между приложением и устройством обработки данных (процессором данных) и включает в себя спецификации и определения команд и ответов приложения. Это позволяет использовать данные и команды, которые будут указаны стандартизированным способом, независимым от радиоинтерфейсов, определенных серией стандартов ИСО/МЭК 18000*.

В настоящем стандарте:

- приведены рекомендации о том, каким образом данные должны представляться в качестве объектов;
- определена структура идентификаторов объектов в соответствии с ИСО/МЭК 9834-1**;
- определены команды, которые поддерживаются для передачи данных между приложением и радиочастотной меткой;
- определены ответы, которые поддерживаются для передачи данных между радиочастотными метками и приложением;
- не определен требуемый синтаксис передачи в ИСО/МЭК 15962, но приведена справочная информация, указанная в приложении А, для обеспечения обратной совместимости с ИСО/МЭК 15961:2004[1)] ***.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты [для датированных ссылок применяют только указанное издание ссылочного стандарта, для недатированных — последнее издание (включая все изменения)]:

ISO/IEC 15961-3, Information technology — Radio frequency identification (RFID) for item management — Data Protocol — Part 3: RFID data constructs [Информационные технологии. Радиочастотная идентификация (РЧИ) для управления предметами. Протокол данных. Часть 3. Конструкции данных радиочастотной идентификации]

ISO/IEC 15962, Information technology — Radio frequency identification (RFID) for item management — Data protocol: data encoding rules and logical memory functions [Информационные технологии. Радиочастотная идентификация (РЧИ) для управления предметами. Протокол данных. Правила кодирования данных и функции логической памяти]

ISO/IEC 19762, Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary [Информационные технологии. Технологии автоматической идентификации и сбора данных (АИСД). Гармонизированный словарь]

[1)] Стандарт отменен. Заменен на ИСО/МЭК 15961-1, ИСО/МЭК 15961-2, ИСО/МЭК 15961-3 и ИСО/МЭК 15961-4.

* *См.* [*10*] — [*18*].

** *См.* [*5*].

*** *См.* [*8*].

Издание официальное

3 Термины, определения и сокращения

3.1 Термины и определения

В настоящем стандарте применены термины по ИСО/МЭК 19762, а также следующие термины с соответствующими определениями.

ИСО и МЭК поддерживают терминологические базы данных для использования в стандартизации по следующим адресам:

- платформа онлайн-просмотра ИСО: доступна по адресу gost.e-ecolog.ru
- Электропедия МЭК: доступна по адресу https:// gost.e-ecolog.ru

3.1.1 **приложение** (application): Компонент программного обеспечения, который выдает команды и получает ответы на команды внутри системы.

3.1.2 **процессор данных** (data processor): Способ реализации процессов, определенных в ИСО/МЭК 15962, включая средство уплотнения данных, средство форматирования, логическую память и блок управления/ответа.

Примечание 1 — В ИСО/МЭК 15691:2004[2)] это называлось «устройством обработки протокола данных» (data protocol processor).

3.2 Сокращения и обозначения

В настоящем стандарте использовано следующее сокращение:

URN — унифицированное имя ресурса (uniform resource name).

Поскольку обозначения «TDS», «Type C» (тип C) и «Type D» (тип D) широко используются в индустрии для обозначения технологических компонентов, как указано в соответствующих стандартах, в настоящем стандарте используются следующие обозначения:

Type C (тип C) — требования по ИСО/МЭК 18000-63*;

Type D (тип D) — требования по ИСО/МЭК 18000-64**;

TDS — требования по стандарту данных радиочастотных меток GS1 EPC***.

4 Соответствие стандартам

4.1 Общие положения

Команды и ответы в настоящем стандарте выражены только в абстрактном синтаксисе, а кодирование передачи (пересмотренная редакция настоящего стандарта 2004 года) больше не требуется. Таким образом, соответствие настоящему стандарту определяется способом кодирования памяти радиочастотных меток, выполняемым согласно требованиям ИСО/МЭК 15962.

Аргументы и поля, содержащиеся в отдельных командах и ответах, определяют, что необходимо учитывать для обеспечения корректности входных данных для процессора данных (data processor) для обеспечения необходимого способа кодирования. Они также определяют то, какое приложение ожидает получить доступ к радиочастотной метке. Благодаря структуре протокола данных команды и ответы, указанные в настоящем стандарте, в значительной степени независимы от конкретных типов радиочастотных меток, известных только процессору данных (data processor) через драйвер радиочастотных меток. Результатом этого является то, что ИСО/МЭК 15962 может указывать требования для обеспечения требуемого способа кодирования, которые не относятся к настоящему стандарту.

Настоящий раздел содержит рекомендации для наилучшего использования при установлении интегрированного канала обмена данными между приложением и радиочастотной меткой.

4.2 Соответствие требованиям

Ожидается, что приложение будет поддерживать те команды и ответы, которые являются значимыми для приложения. Для каждой команды, относящейся к приложению, все ее параметры должны учитываться в процессе обмена данными между приложением и процессором данных (data processor).

---

2) Отмененный стандарт. Заменен на ИСО/МЭК 15961-1, ИСО/МЭК 15961-2, ИСО/МЭК 15961-3 и ИСО/МЭК 15961-4.

* *См.* [*16*].

** *См.* [*17*].

*** *См.* [*22*].

В частности, стандарты приложения должны учитывать множество аргументов, используемых командами, в соответствии с разделом 7 [например, «блокировка объекта» (Object-Lock), «параметр уплотнения» (Compact-Parameter)]. Они также должны определять требования к содержимому данных, записываемых в радиочастотную метку, а также необходимым процессам, которые может вызывать процессор данных (data processor) для обеспечения требуемого способа кодирования.

4.3 Соответствие процессора данных (Data Processor)

Процессор данных является, по сути, реализацией ИСО/МЭК 15962. В зависимости от области применения процессора данных (Data Processor) (начиная от специфического для конкретной отрасли промышленности до универсального для всего радиочастотного протокола данных) существуют различные аргументы, включенные в команды, которые могут обрабатываться разными способами (например, данные могут быть идентифицированы полным идентификатором объекта или производным относительным идентификатором объекта). Настоящий стандарт не налагает никаких ограничений на структуру процессора данных (data processor), кроме необходимости поддерживать все функциональные возможности, указанные аргументами в командах, необходимых для обеспечения надлежащего формата кодирования.

5 Модель протокола

В настоящем стандарте применяют модель протокола по ИСО/МЭК 15962.

6 Соглашения о представлении

6.1 Представление команд, ответов и аргументов

6.1.1 Общие положения

Команды и ответы определяются в формате прямоугольника, как показано на рисунке 1.

Каждая команда или ответ содержит упорядоченный список полей или аргументов. При необходимости после имени поля/аргумента приводится указание на тип данных и краткое описание. Для обозначения полей, содержимое которых определяется ограниченным подмножеством допустимых значений, используется курсив.

Примечание — Ничто из указанного не применяется к оригинальным командам или ответам, представленным в абстрактном синтаксисе АСН.1 (ASN.1)*, как в ИСО/МЭК 15961:2004**.

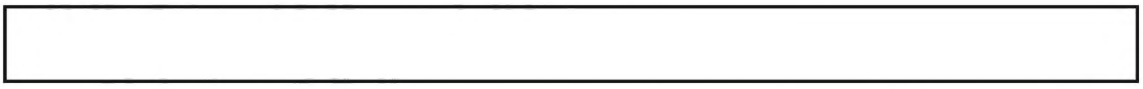


Рисунок 1 — Формат прямоугольника для команд, ответов и составных аргументов

6.1.2 Типы данных

Приведенные ниже типы данных используются в командах и ответах:

- BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ): аргументы, которые могут иметь значения TRUE (ИСТИНА) или FALSE (ЛОЖЬ);
- BIT STRING (ДВОИЧНАЯ СТРОКА): последовательность битов;
- BYTE (БАЙТ): целочисленная переменная с возможными значениями от 0 до 255, обычно выраженная в виде шестнадцатеричного значения от 00_{16} до FF_{16};
- BYTE STRING (СТРОКА БАЙТОВ): последовательность байтов (эквивалент СТРОКИ ОКТЕТОВ);
- EBV-8: двоичный метод кодирования чисел переменной длины в одном поле с использованием начального бита индикатора, предшествующего 7-битовому значению. Завершающий компонент EBV-8 должен начинаться с 0, а все предыдущие компоненты начинаются с 1. Например, $6910=01000101_2$, в то время как $369_{10}=\mathbf{1}01110001_2$, в EBV-8 будет записываться как **1**0000010 **0**1110001 (т. е. разделение на 7-битовые строки с добавлением индикатора в качестве префикса). Единственное требование для использования кода EBV-8 в команде состоит в том, где этот тип значения возвращается в ответе по радиоинтерфейсу;

---

* *АСН.1 (ASN.1) — абстрактная синтаксическая нотация версии один (Abstract Syntax Notation One).*

** *См.* [*8*].

- HEXADECIMAL ADDRESS (ШЕСТНАДЦАТЕРИЧНЫЙ АДРЕС): область (памяти радиочастотной метки), выраженная шестнадцатеричным значением;

- INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ): целочисленное значение может принимать в качестве значения любое целое число. В контексте настоящего стандарта полагается, что все принимаемые значения — положительные;

- OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА): идентификатор объекта, как определено в 6.2.1.

6.2 Представление идентификатора объекта в интерфейсе приложения

6.2.1 Структура идентификатора объекта по ИСО/МЭК 8824-1

Настоящий стандарт использует тип OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА), как указано в ИСО/МЭК 8824-1*, с идентификаторами, присваиваемыми в соответствии с ИСО/МЭК 9834-1**. Используется древовидное представление регистрации с общим включенным корневым узлом (ИСО/МЭК 9834-1**), серией дуг от каждого узла, с добавлением новых дуг, необходимых для определения конкретного объекта (см. рисунок 2). Таким образом, организация, ответственная за конкретный узел:

- имеет определенный набор дуг для идентификации;
- может управлять расположением дуг, относящихся к своему узлу, независимо от других объектов;
- гарантирует уникальность по отношению ко всем другим дугам в древовидном представлении регистрации.

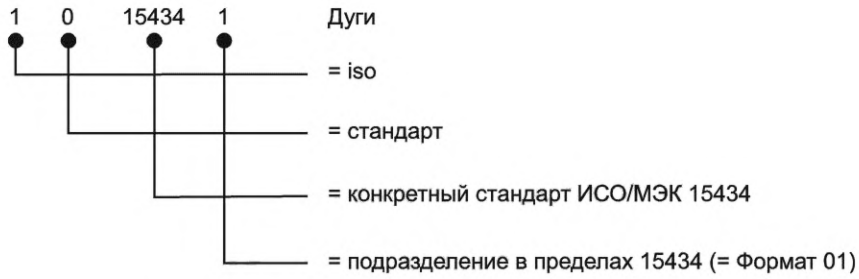


Рисунок 2 — Пример дуг идентификатора объекта

Единственные верхние дуги, разрешенные для всех идентификаторов объектов, приведены в таблице 1.

Таблица 1 — Дуги верхнего уровня идентификатора объекта

Имя дуги идентификатора	Числовое значение
itu-t	0
iso	1
joint-iso-itu-t	2

Примечание 1 — Любой стандарт ИСО/МЭК, такой как настоящий стандарт, имеет идентификаторы объекта под дугой ISO верхнего уровня.

Дуга второго уровня управляется соответствующей организацией, назначенной для дуги верхнего уровня. Текущий список дуг верхнего и второго уровней приводится в ИСО/МЭК 15961-3.

Дуга третьего уровня управляется системой или организацией, определенной для дуги второго уровня. Иногда (вместо нее) ею является регистрирующий орган. Иерархическая структура выстраивается до тех пор, пока объект не будет идентифицирован как уникальный. Процедура именования идентификаторов объектов гарантирует, что каждый объект уникален в пределах «родительской» дуги и что каждая «родительская» дуга уникальна на своем предыдущем уровне, вплоть до трех верхних дуг.

Примечание 2 — Эта структура позволяет однозначно кодировать идентификаторы объектов из разных доменов (например, открытых и закрытых систем) в памяти радиочастотной метки.

---

* *См.* [*2*].

** *См.* [*5*].

Протокол данных радиочастотной идентификации использует три формы идентификатора объекта:

- идентификатор объекта (Object-Identifier): эта полная структура используется для связи между приложением и процессором данных (data processor), определяемым областью применения настоящего стандарта;

- корневой идентификатор объекта (Root-OID): корневой идентификатор объекта является общей частью набора закодированных идентификаторов объектов. Он действует как общий префикс для относительного идентификатора объекта, закодированного в памяти радиочастотной метки. Эта структура особенно важна для кодирования данных в радиочастотной метке в приложениях, которые требуют множество данных из общего словаря данных для кодирования. Корневой идентификатор объекта (Root-OID) либо явно закодирован, либо объявлен в соответствии с правилами кодирования в ИСО/МЭК 15962;

- относительный идентификатор объекта (Relative-OID): эта структура используется совместно с корневым идентификатором объекта (Root-OID) (см. ниже) для связи между приложением и процессором данных (data processor), определяемым областью действия настоящего стандарта. Эти структуры используют в ситуациях, когда к набору идентификаторов объектов, которые должны быть записаны в память радиочастотной метки, применяется общий корень. Например, если все идентификаторы объектов имеют общий корень 1 0 15961 12, области кодирования могут быть сохранены в памяти радиочастотной метки при условии, что этот общий корневой идентификатор объекта (Root-OID) не должен кодироваться для каждого идентификатора объекта. Относительный идентификатор объекта (Relative-OID) является суффиксом для общего корневого идентификатора объекта (Root-OID), который либо закодирован, либо объявлен каким-либо другим способом.

Примечание 3 — Термин «идентификатор объекта» (Object-Identifier) может применяться к относительному идентификатору объекта (Relative-OID), за исключением тех случаев, когда требуется их четкое разделение, а также в тех случаях, когда команды и ответы представлены в абстрактной форме (см. раздел 10).

6.2.2 Представление идентификатора объекта в соответствии с ИСО/МЭК 8824-1

Если идентификатор объекта (Object-Identifier) представлен в соответствии с ИСО/МЭК 8824-1[*], то пробелы вставляются между каждой дугой следующим образом:

1 0 15961 12 1

Примечание — В представлении ACH.1 это может быть выражено более формально: {iso(1) standard(0) rfid-data-protocol(15961) iata(12) baggage-id(1)}.

6.2.3 Представление идентификатора объекта в виде унифицированного имени ресурса (Uniform Resource Name — URN)

Идентификатор объекта (Object-Identifier) может быть также представлен в виде URN на базе формата, основанного на IETF RFC 3061[**], с разделителем в виде десятичной точки между каждой дугой:

urn:oid:1.0.15961.12.1

6.3 Байтовая нотация

6.3.1 Байт: базовый элемент для 8-битового кодирования

Настоящий стандарт поддерживает двоичные, 6-битовые, 7-битовые, 8-битовые блоки данных, а также любые другие пользовательские данные, длина которых превышает 8 бит на знак. Общей единицей кодирования является 8-разрядный байт (также известный как октет).

Двоичные данные должны заполняться начальными нулевыми битами, пока двоичное значение не будет выровнено по октету; 7-битовые данные должны быть представлены в виде байтов с 8-м битом (см. 6.3.2), установленным в нулевое значение. Данные, превышающие 8 бит, должны кодироваться в нескольких байтах.

8-битовый байт представлен двумя шестнадцатеричными значениями с использованием знаков 0—9 и A—F.

6.3.2 Упорядочение битов

В каждом байте самым старшим значащим битом является бит 8, а самым младшим значащим битом является бит 1. Соответственно, весовой коэффициент, выделенный каждому биту, приведен в таблице 2.

---

[*] *См.* [*2*].

[**] *См.* [*23*].

Таблица 2 — Последовательность битов в 8-битовом байте

Значение бита	Бит 8	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1
Весовой коэффициент	128	64	32	16	8	4	2	1

6.3.3 Преобразование байтов

8-битовое значение преобразуется в два шестнадцатеричных знака. Для определения первого шестнадцатеричного знака используются бит 8, бит 7, бит 6 и бит 5 с весовыми коэффициентами 8, 4, 2 и 1 соответственно. Для определения второго шестнадцатеричного знака используются бит 4, бит 3, бит 2 и бит 1 с весовыми коэффициентами 8, 4, 2 и 1 соответственно.

7 Обработка команд и ответов приложения

7.1 Общие данные

Со времени публикации первых изданий ИСО/МЭК 15961 (ИСО/МЭК 15961:2004*) и ИСО/МЭК 15962 стало понятно, что ряд конфигураций некоторых устройств не требует наличия функции кодирования передачи данных в их конфигурации для надлежащей поддержки функции протокола передачи данных. Конфигурации таких устройств имеют схожие характеристики в том смысле, что они могут поддерживать все внутренние процессы (например, устройства печати-кодирования, портативные устройства считывания радиочастотной идентификации и программные пакеты), но конфигурации других устройств могут иметь схожие характеристики. Эти характеристики включают в себя полный процесс кодирования, который поддерживает функциональные возможности команды приложения и процессы кодирования в логической памяти и/или процессы декодирования из логической памяти в ответы приложения.

В таких устройствах появляются два ненужных процесса, добавляющих сложности и увеличивающих время исполнения из-за необходимости выполнения кодирования во время передачи, результат которого должен быть моментально декодирован и таким образом отменен. Как часть преобразования ИСО/МЭК 15961:2004* в стандарт, состоящий из нескольких частей, настоящий стандарт, являющийся одной из этих частей, теперь поддерживает прямое кодирование передаваемых данных.

Существует два принципиально разных способа реализации поддержки команд и ответов, определенных в разделе 10.

7.1.1 Вариант А: прямолинейный процесс

В устройствах или программном обеспечении, поддерживающих непосредственный ввод данных для команд, которые совместимы с настоящим стандартом, а также поддерживают процессы кодирования по ИСО/МЭК 15962, кодирование передачи данных не требуется. Поэтому важно обеспечить базовую функциональность команд и ответов, для этого должны соблюдаться правила, определенные для идентификаторов объектов (см. 7.3.3) и аргументов команды и ответа (см. 7.4).

Для этой сквозной обработки протокола данных допустимы различные формы реализации, включая непосредственный ввод из форм или более прямую передачу данных из хост-систем. В настоящем стандарте отсутствуют ограничения на этот процесс.

7.1.2 Вариант В: кодирование передачи данных

Допустимо использовать исходное кодирование передачи данных (см. приложение А) в случаях, если устройство или программное обеспечение поддерживает только настоящий стандарт или поддерживает только процессы кодирования по ИСО/МЭК 15962. Для лучшей интеграции с системами, поддерживающими такие интерфейсные механизмы, допустимо использовать альтернативное кодирование передачи данных, включая бинарное кодирование, определенное в ИСО/МЭК 24791-5**. В свою очередь, это означает, что структуры команд и ответов, определенные в разделе 7, также должны рассматриваться как абстрактные или функциональные определения требований к надлежащим образом закодированным данным в памяти радиочастотной метки или считывать и интерпретировать данные из радиочастотной метки.

---

* *См.* [*8*].

** *См.* [*21*].

7.2 Информация, связанная с системой кодирования в командах

7.2.1 Индивидуализированный идентификатор (Singulation-Id)

Индивидуализированный идентификатор (Singulation-Id) (ранее определенный в ИСО/МЭК 15961:2004* как TagId) предоставляется драйвером радиочастотных меток (Tag Driver) и однозначно идентифицирует радиочастотную метку в течение периода транзакции данных.

В протоколе данных идентификатор Singulation-Id должен иметь длину до 255 байт и действовать как ссылка на файл для логической памяти и, в свою очередь, обеспечивать связь «один к одному» с картой логической памяти самой радиочастотной метки. Идентификатор Singulation-Id должен основываться на одном из следующих вариантов (выбор которого определяется драйвером радиочастотных меток и протоколом радиоинтерфейса):

a) полностью уникальный идентификатор предмета (Unique Item Identifier), запрограммированный в радиочастотной метке, как указано в серии стандартов ИСО/МЭК 18000**;

b) идентификатор, связанный с данными, например уникальный идентификатор предмета (Unique Item Identifier, UII), обеспечивающий уникальность в пределах области применения системы управления объектами. Для того, чтобы установить идентификатор Singulation-Id, требуется считать идентификатор UII;

c) виртуальный или сеансовый идентификатор на основе временного интервала или другой функции, управляемой протоколом радиоинтерфейса;

d) комбинации вариантов b) и c), например виртуальный идентификатор Singulation-Id по радиоинтерфейсу, только требующий возврата идентификатора, связанного с данными, в качестве ответа.

7.2.2 Идентификатор семейства приложений (AFI)

Идентификатор семейства приложений (AFI) (ранее определенный в ИСО/МЭК 15961:2004* как ApplicationFamilyId) относится к специальным идентификаторам, которые обеспечивают выборочную адресацию радиочастотных меток. Это может поддерживаться механизмом в рамках радиоинтерфейса. Структура кодов, представленная ниже, не противоречит серии стандартов для смарт-карт, поскольку они также поддерживают AFI. Значение идентификатора AFI для радиочастотной идентификации для управления предметами должно иметь размерность один байт, значение которого определяется в ИСО/МЭК 15961-3 следующим образом:

- если это шестнадцатеричное значение находится в диапазоне от 00_{16} до $0F_{16}$, то это кодовое значение определяет приложение, относящееся к закрытым системам, как определено в ИСО/МЭК 15961-3. Список кодовых значений обновляется в соответствующем регистре конструкций данных радиочастотной идентификации;

- если это шестнадцатеричное значение находится в диапазоне от 90_{16} до CE_{16}***, то это кодовое значение определяет приложение, относящееся к закрытым системам, как определено в ИСО/МЭК 15961-3. Список кодовых значений обновляется в соответствующем регистре конструкций данных радиочастотной идентификации;

- шестнадцатеричное значение CF_{16} зарезервировано как код расширения для кодовых значений идентификатора AFI, состоящего из нескольких байтов.

Идентификаторы AFI должны храниться в радиочастотной метке в той или иной форме или, альтернативно, могут быть определены службами радиоинтерфейса, если они достаточно специфичны.

Если идентификатор AFI не поддерживается классом радиочастотной метки, а службы радиоинтерфейса также не предоставляют данных о AFI, то это кодовое значение в информации о системе должно принимать шестнадцатеричное значение 00_{16}.

7.2.3 Идентификатор формата хранения данных (DSFID)

Идентификатор формата хранения данных (DSFID) [ранее определенный в ИСО/МЭК 15961:2004* как формат хранения (StorageFormat)] относится к группе идентификаторов, которые позволяют эффективно кодировать данные, записываемые в память радиочастотные метки. Значение идентификатора DSFID в системе радиочастотной идентификации для управления предметами состоит из одного или нескольких байтов. Он кодирует метод доступа (Access-Method) и формат данных (Data-Format), определения которых представлены ниже. Метод доступа (Access-Method) кодируется в битах 8 и 7 идентификатора DSFID, причем последние пять бит (биты 5—1) кодируют формат данных (Data-Format). Бит 6 определяет дополнительные функции, поддерживаемые некоторыми типами радиочастотных меток.

---

* *См.* [*8*].

** *См.* [*10*] — [*18*].

*** *В ИСО/МЭК 15961-1 ошибочно указано* $0F_{16}$.

Если метод доступа (Access-Method) имеет значение от 4 до 15, расширения для идентификатора DSFID реализуются процессором данных (data processor).

Примечание — Ни одно из этих значений еще не присвоено, и такие присвоения будут сделаны только посредством пересмотра или внесения поправок в настоящий стандарт.

Если формат данных (Data-Format) имеет значение от 32 до 287, как определено в ИСО/МЭК 15961-3, расширения для идентификатора DSFID реализуются процессором данных (data processor). Поскольку формат данных (Data-Format) зарегистрирован как часть регистра конструкций данных в ИСО/МЭК 15961-2*, это расширение происходит независимо от изменений в настоящем стандарте.

7.2.4 Метод доступа (Access-Method)

Метод доступа (Access-Method) определяет способ размещения данных в памяти радиочастотной метки и доступ к ним из радиочастотной метки. Значение метода доступа должно храниться в памяти радиочастотной метки или может определяться службами радиоинтерфейса, если это можно сделать однозначно. Метод доступа (Access-Method) определяется значением бита, при этом должны применяться следующие кодовые значения:

0 — No-Directory (без каталога): этот метод доступа предоставляет самый простой набор правил кодирования. Основное правило — связать объект (Object) с идентификатором объекта (Object-Identifier) и поддерживать его соответствующими синтаксическими компонентами для создания набора данных. Наборы данных (Data-Sets) объединяются для создания непрерывной последовательности закодированных байтов в карте логической памяти;

1 — Directory (каталог): структура Directory (каталог) поддерживает все правила кодирования структуры No-Directory (без каталога), но дополнительно поддерживает кодирование каталога непосредственно, сведения о кодировании содержатся в старших значениях адреса в карте логической памяти в радиочастотной метке. Когда используется структура Directory (каталог), любая команда, стремящаяся выборочно считывать данные, может сначала получить доступ к каталогу, чтобы найти адрес памяти начала набора данных (Data-Set), а затем перейти в это место, чтобы считать байты, которые представляют собой кодирование набора данных.

Структура Directory (каталог) может быть записана в радиочастотную метку позднее, чем первоначальные закодированные данные;

2 — Packed-Objects (упакованные объекты): структура Packed-Objects (упакованные объекты) была введена в первой редакции ИСО/МЭК 15962 и настоящей редакции данного стандарта. Схема кодирования принципиально отличается от ранее описанных, поскольку она использует набор объектов (Objects) и их идентификаторов (Object-Identifiers), а затем кодирует их в индексированную структуру данных, обеспечивающую одновременное сжатие и кодирование данных. Схема кодирования Packed-Objects (упакованные объекты) требует наличия таблицы правил кодирования для каждого формата данных (Data-Format), которая вызывает конкретные схемы уплотнения для отдельных элементов. Схемы уплотнения являются стандартными независимо от формата данных. Схема Packed-Objects (упакованные объекты) требует более сложных правил кодирования и декодирования данных в отличие от схем No-Directory (без каталога) и Directory (каталог), однако она обеспечивает значительную эффективность кодирования по сравнению с базовой структурой No-Directory (без каталога), для которой требуется осуществить кодирование нескольких идентификаторов объектов (Object-Identifiers). Это позволяет уменьшить объем памяти, используемой радиочастотной меткой, а также может уменьшить размер сообщения, передаваемого по радиоинтерфейсу в ответ на команду считывания. Для реализации метода доступа (Access-Method) данного типа требуется определить таблицу упорядоченных относительных идентификаторов объекта (Relative-OIDs). Данный метод доступа может применяться только как альтернатива методам доступа No-Directory (без каталога) или Directory (каталог) для определенной радиочастотной метки. Тем не менее в одном приложении допустимо использовать радиочастотные метки с любым из методов доступа;

3 — Tag-Data-Profile (профиль данных радиочастотных меток): схема Tag-Data-Profile (профиль данных радиочастотных меток) предназначена для поддержки сообщений с фиксированной структурой и применяется обычно там, где необходимо кодировать несколько идентификаторов объектов (Object-Identifiers) вместе с их объектами (Objects). Фиксированная структура сообщений лучше всего подходит для приложений, которые являются достаточно однородными и дополнительно имеют согласованное требование для кодирования одинаковых идентификаторов объектов в памяти всех радиочастотных меток для одного приложения. Для каждой структуры Tag-Data-Profile (профиль данных радиочастотных меток) требуется регистрация;

* *См.* [9].

4 — Multiple-Records (блочные записи): процесс кодирования Multiple-Records (блочные записи) представляет собой структуру кодирования методов доступа, перекрывающую все ранее рассмотренные: No-Directory (без каталога), Packed-Objects (упакованные объекты и Tag-Data-Profile (профиль данных радиочастотных меток) для множества сущностей этих методов доступа к памяти (Access Methods), и представляет собой их комбинацию, позволяющую осуществлять кодирование в той же логической памяти. Это достигается за счет введения закодированных заголовка блочной записи (MR-header) и преамбулы (preamble) для каждой записи. Это позволяет кодированию отдельных записей полностью соответствовать свойственной им системе доступа. Это также позволяет кодировать один и тот же идентификатор объекта в отдельных записях с общим правилом, поддерживающим список одного и того же элемента данных в одной записи. Существует три основных класса приложений, поддерживаемых системой доступа, и их также допускается смешивать. Один класс позволяет различным форматам данных и владельцам данных использовать одну и ту же радиочастотную метку, но под контролем оригинального владельца радиочастотной метки. Другой поддерживает временную последовательность записей истории одного и того же типа, которую необходимо повторить. Третий основной идентификатор класса предназначен для поддержки иерархически связанных записей, например для доставки в цепи поставок или спецификации типов материалов конструкции;

5—15 — зарезервированные значения для будущих схем кодирования, которые должны быть определены в ИСО/МЭК 15962 и вызываться по правилам для многобайтового идентификатора DSFID. Возможно, некоторые из новых методов доступа (Access-Methods) могут быть неотъемлемыми схемами кодирования в блочной записи (Multiple-Records). Это можно определить только при разработке схем кодирования.

Там, где можно сделать выбор, могут помочь следующие рекомендации:

- структура No-Directory (без каталога) лучше подходит для радиочастотных меток с малой емкостью памяти, потому что сама структура Directory (каталог) накладывается на данные, которые необходимо закодировать. Она также лучше подходит для того, чтобы закодировать несколько идентификаторов объектов (Object-Identifiers), чтобы функция непрерывного считывания передавала все кодирование (или, по крайней мере, достаточный объем из закодированных байтов), чтобы дать возможность провести структурный анализ байтов для поиска требуемого идентификатора объекта и соответствующего объекта;

- структура Directory (каталог) явно подходит лучше, когда условия отличаются от условий, подходящих для структуры No-Directory (без каталога). Кроме того, она лучше подходит для приложений, которые требуют выборочного считывания, записи или изменения одного или нескольких идентификаторов объектов (Object-Identifiers) из множества. В ситуации этого типа используется дополнительное чтение для передачи сведений о директории в процессор данных (data processor). Это считывание скорее всего должно быть сбалансировано для сокращения времени чтения выбранных идентификаторов объектов;

- структура Packed-Objects (упакованные объекты) требует использования более сложных процессов кодирования и декодирования, но она позволяет значительно улучшить эффективность кодирования по сравнению с базовой структурой No-Directory (без каталога). Таким образом, она лучше подходит для приложений, где требования к кодированию относительно высоки по сравнению с объемом памяти радиочастотных меток, используемых в приложении;

- схема Tag-Data-Profile (профиль данных радиочастотных меток) лучше всего подходит для однородных систем (например, вертикально интегрированная цепь поставок, где все поставщики известны всем клиентам одновременно), и все стороны соглашаются с тем, что каждый идентификатор объекта (Object-Identifier) должен быть закодирован и согласован с общей структурой и форматом данных для каждого объекта (Object).

Поскольку метод доступа (Access-Method) должен быть указан приложением, следует обеспечить возможность измерения передачи данных, которая имитирует структуру данных Directory (каталог) и структуру данных в структуре No-Directory (без каталога). Такой тест может применяться для определения оптимального решения при выборе между этим и другими методами доступа, с учетом разнообразия и длины данных, а также типичных реализаций считывания/записи, ожидаемых для приложения.

После того как для радиочастотной метки был выбран соответствующий метод доступа, приложению нет необходимости определять, как должны осуществляться считывание, запись или каким образом структурированы байты в логической памяти. Это функция процессора данных (data processor) и драйвера радиочастотных меток.

7.2.5 Формат данных (Data-Format)

Формат данных (Data-Format) определяет корневой идентификатор объекта (Root-OID) из идентификаторов объекта (Object-Identifiers), которые хранятся в памяти радиочастотной метки. Формат данных позволяет объектам данных быть простыми относительными идентификаторами объектов (Relative-OID), тем самым более эффективно использовать пространство для кодирования или ограничивать данные одним классом.

Полные идентификаторы объектов [т. е. с другим корневым идентификатором объекта (Root-OID), который определяется форматом данных (Data-Format)] все еще могут быть закодированы в памяти радиочастотной метки, позволяя кодировать данные из разных прикладных областей, хотя и не на том же уровне эффективности кодирования.

Значение формата данных должно храниться в радиочастотной метке или может быть определено службами радиоинтерфейса, если это можно сделать однозначным образом. При определении с помощью протокола радиоинтерфейса должна существовать взаимно-однозначная связь с типом радиочастотной метки и значением формата данных. Значение формата данных в системе радиочастотной идентификации для управления предметами должно быть целочисленным значением в диапазоне от 0 до 287, указанным в ИСО/МЭК 15961-3. В настоящем стандарте применимы следующие значения:

0 — Not-Formatted (без отформатирования): используется для неотформатированных радиочастотных меток или еще не отформатированых согласно настоящему стандарту;

1 — Full-Featured (полнофункциональный): этот формат данных поддерживает любой тип формата данных, в котором используется полный идентификатор объекта. Его основная цель — включить неоднородные данные (т. е. находящиеся в разных словарях данных) для кодирования в одной радиочастотной метке. Например, он может использоваться для кодирования данных из разных приложений типа «закрытая система», используя зарегистрированные идентификаторы объектов по ИСО/МЭК 9834-1* для любого приложения;

2 — Root-Oid-Encoded (закодированный корневой идентификатор объекта): этот формат данных используется, когда все данные радиочастотной метки имеют общий корневой идентификатор объекта (Root-OID), который не соответствует одному из определенных корневых идентификаторов объекта, связанному с форматом данных, зарегистрированным по правилам ИСО/МЭК 15961-2**.

Корневой идентификатор объекта (Root-OID) для формата данных 2 должен быть напрямую закодирован в радиочастотной метке с использованием соответствующих корневых дуг. Каждый объект кодируется в радиочастотной метке с использованием относительного идентификатора объекта (Relative-OID), представляющего оставшиеся дуги нижнего уровня;

3—28 — указанные кодовые значения применяют к приложениям типа «открытые системы», зарегистрированным в ИСО/МЭК 15961-2**. Список кодовых значений обновляется в соответствующем регистре конструкций данных радиочастотной идентификации;

29 — указанное кодовое значение применяют к закрытым системным данным, где закодированные данные соответствуют правилам настоящего стандарта и ИСО/МЭК 15962;

30 — указанное кодовое значение применяют к закрытым системным данным, где правила кодирования не соответствуют ИСО/МЭК 15962. Оно применимо для приложений, которые мигрируют из других правил кодирования, позволяя отличать данные и надлежащим образом кодировать их. Оно также используется до перехода радиочастотных меток из закрытого системного приложения в открытое системное приложение;

31 — указанное значение не назначено ни одному приложению, так как его двоичный эквивалент используется для указания расширения к формату данных в диапазоне от 32 до 287;

32—287 — указанные значения зарезервированы для открытых системных приложений, зарегистрированных в соответствии с ИСО/МЭК 15961-2**, Предполагается, что такие приложения задействуют механизм расширения в соответствии с требованиями ИСО/МЭК 15962 для многобайтового идентификатора DSFID.

7.3 Подготовка основных объектов и других аргументов основных приложений

7.3.1 Общие положения

Это начальный процесс для обеспечения подготовки объектов данных в формате, совместимом с настоящим стандартом.

---

* *См.* [*5*].

** *См.* [*9*].

Примечание — Признано существование протоколов, основанных на передаче сообщений, и синтаксиса для прикладных стандартов для существующих АИСД (AIDC), которые отличаются от протоколов настоящего стандарта, основанных на передаче объектов. Возможно реализовать преимущества протокола, основанного на передаче объектов, и поддерживать совместимость с системами, основанными на передаче сообщений, с помощью преобразований между этими двумя форматами (см. приложение В).

Формат выходных данных, приводимый в последующих подразделах, должен соответствовать формату входных данных для процессора данных (data processor). Однако механизм достижения этого соответствия в настоящих подразделах не рассматривается. Требование состоит в том, чтобы присвоить идентификатор объекта каждому объекту, используя словарь данных, соответствующий стандарту приложения.

7.3.2 Основная модель

На рисунке 3 приведена модель потока данных для подготовки каждого идентификатора объекта (Object-Identifier) и связанного с ним объекта данных (data Object). Каждый тип данных приложения, и особенно те, которые охватываются настоящим стандартом, имеет словарь данных (data dictionary) или список объектов данных (data Objects) или элементов данных (data element).

Любой такой словарь данных может совершенствоваться в течение определенного периода времени, и поддержка такого словаря данных не зависит от настоящего стандарта. Словарь данных обычно состоит из кодированного списка (числового, алфавитного, алфавитно-цифрового и т. д.) объектов данных и их спецификации для использования в предметной области стандарта приложения. ИСО/МЭК 15961-3 и связанный с ним регистр конструкций данных радиочастотной идентификации определяют идентификаторы объектов, которые относятся к приложениям.



Рисунок 3 — Модель потока данных: подготовка базовых объектов

7.3.3 Идентификатор объекта (Object-Identifier)

Идентификатор объекта (Object-Identifier) должен быть предоставлен системой приложения в виде серии дуг, как определено в 6.2, включая все идентификаторы объектов (Object-Identifiers), связанные с установленными форматами данных (DataFormats).

Идентификатор объекта (Object-Identifier) должен быть представлен как (полный) идентификатор объекта в командах приложения, если только система, зависящая от конкретного приложения, не поддерживает все соответствующие относительные идентификаторы объектов (Relative-OID). По возможности процессор данных (data processor) преобразует (полный) идентификатор объекта (Object-Identifier) в относительный идентификатор объекта (Relative-OID) для более эффективного кодирования в радиочастотной метке на основе формата данных, полученных из радиочастотной метки.

7.3.4 Установка соотношений идентификаторов объектов

Синтаксис, ориентированный на передачу сообщений, может использовать рекурсивные или циклические методы для создания повторяющихся последовательностей связанных данных (например, отдельных значений количества и номеров партий, связанных с разными кодовыми значениями продукции). При анализе полного сообщения синтаксис определяет граничные точки, чтобы атрибуты были надлежащим образом связаны с основным кодовым значением.

Примечание — При использовании объектно-ориентированной системы (Object-based system) (такой, как протокол данных в настоящем стандарте и ИСО/МЭК 15962), работающей на базовом уровне, существует риск создания ложных ссылок (например, кодовое значение продукции А может быть связано с количеством продукции В). Проблема может быть преодолена с использованием одного из методов, описанных в приложении С.

7.3.5 Объект (Object)

7.3.5.1 Общие положения

Приложение предоставляет данные в форме объекта (Object). Его значение представлено в байтовой форме и связано с определением идентификатора объекта (Object-Identifier), предоставляемым словарем данных приложения. Конкретные рекомендации о конкретной интерпретации приводятся в следующих подпунктах.

7.3.5.2 Основная информация о 8-битовом наборе знаков

Многие бизнес-приложения используют набор знаков по ИСО/МЭК 8859-1* или по ИСО/МЭК 646**. Таким образом, интерпретация (или представление) данных известна всем участникам системы. Она включает в себя как конкретного отправителя, так и получателя, даже если они не известны напрямую друг для друга, как это зачастую бывает в открытых системах автоматической идентификации и сбора данных (АИСД, AIDC). Если для обработки набора знаков по ИСО/МЭК 8859-1* настроены текстовое приложение и приложение для браузера, большая часть содержимого данных может отображаться так, как и предполагалось. Если же информационное наполнение имеет определенную интерпретацию, приложение радиочастотной идентификации должно точно определить между участниками, каким образом будут интерпретироваться конкретные данные. Это может быть достигнуто путем публикации словаря данных с использованием идентификатора объекта в качестве ссылки.

7.3.5.3 Поддержка для наборов знаков по ИСО/МЭК 10646

ИСО/МЭК 10646*** поддерживает глифы знаков (визуальные представления знаков) всех наборов знаков. Одним из вариантов поддержки данных в любом другом конкретном наборе знаков является их предварительная обработка путем преобразования в ИСО/МЭК 10646*** с использованием инструментов сопоставления, а затем их уплотнение для кодирования в формате UTF-8 (как определено в ИСО/МЭК 10646***). Любые данные, непосредственно определенные в ИСО/МЭК 10646***, также должны кодироваться в формате UTF-8, чтобы уменьшить число байтов, необходимых для хранения объекта в карте логической памяти.

7.3.5.4 Поддержка защищенных или зашифрованных данных

Все формы защищенных данных, включая зашифрованные данные, должны быть созданы приложением до передачи в качестве объекта в процессор данных (data processor). Это осуществляется по двум причинам:

- требуется, чтобы безопасность данных обеспечивалась в приложении и изменялась независимо от процессора данных;
- позволяет процессору данных обрабатывать все данные аналогичным образом независимо от того, зашифрованы они или нет.

Тот факт, что данные зашифрованы, может быть доведен до сведения всего сообщества пользователей системы, однако фактический метод шифрования может быть ограничен отправителем и предполагаемым получателем.

Более подробная информация приведена в 7.6.

7.3.6 Параметр уплотнения (Compact-Parameter)

Включение Compact-Parameter (параметра уплотнения) в команду должно определять, будут ли данные приложения уплотняться процессором данных (data processor). Параметр уплотнения — это целочисленное значение, в котором применяются следующие кодовые значения:

0 — Application-Defined (определено приложением): объект (Object) не обрабатывается с помощью правил уплотнения данных в ИСО/МЭК 15962 и остается неизменным при сохранении на карте логической памяти радиочастотной метки;

1 — Compact (уплотнение): для этого необходимо использовать основные правила уплотнения ИСО/МЭК 15962 для максимально компактного уплотнения объекта, чтобы уменьшить число байтов, необходимых для карты логической памяти;

2 — UTF8-Data (данные в формате UTF-8): это означает, что объект был внешне преобразован с использованием набора кодированных знаков по ИСО/МЭК 10646***, в формате кодирования UTF-8. Объект не обрабатывается с помощью правил уплотнения данных ИСО/МЭК 15962 и остается неизменным для передачи на карту логической памяти;

3 — Pack-Objects (упакованные объекты): это означает, что набор объектов должен быть закодирован с использованием схемы кодирования PackedObject (упакованные объекты) и идентифицирован

---

* *См.* [*4*].

** *См.* [*1*].

*** *См.* [*6*].

с помощью соответствующего метода доступа (Access-Method). Набор объектов не должен быть уплотнен с использованием основных правил уплотнения ИСО/МЭК 15962, но с использованием правил кодирования упакованного объекта;

4 — Tag-Data-Profile (профиль данных радиочастотных меток): это означает, что набор объектов должен быть закодирован с использованием схемы кодирования Tag Data Profile и идентифицирован с помощью соответствующего метода доступа (Access-Method). Несмотря на то, что набор схем уплотнения идентичен основным правилам уплотнения по ИСО/МЭК 15962, таблица идентификаторов профиля (ProfileIDTable) указывает на конкретную схему уплотнения. Должна быть указана спецификация таблицы идентификаторов профиля;

5 — Monomorphic-UII (мономорфный идентификатор UII): это означает, что один объект, определяющий идентификатор UII, должен быть закодирован с использованием правил уплотнения, определенных идентификатором AFI. Поскольку данное правило применяется к идентификатору UII, этот параметр уплотнения применяется только к одному объекту в банке памяти.

Этот параметр уплотнения определен явно, поскольку для некоторых радиочастотных меток кодирование объектов не зависит от кодирования идентификатора AFI, поэтому процессор данных (data processor) должен получить доступ к информации о параметре уплотнения из регистра конструкций данных, как определено в ИСО/МЭК 15961-2*;

6—13 — зарезервированы для будущего определения.

14 — De-Compacted-Monomorphic-UII (разуплотненный мономорфный идентификатор UII): это означает, что объект в ответе был разуплотнен с использованием правил, определенных с помощью идентификатора AFI, назначенного для определенного мономорфного идентификатора UII.

Этот параметр уплотнения данных определен явно, потому что мономорфный идентификатор UII не имеет связанный идентификатор объекта, закодированный в радиочастотной метке. Идентификатор объекта, определенный с помощью идентификатора AFI, должен быть добавлен в ответ. Процессор данных (data processor) должен получить доступ к информации об идентификаторе объекта из регистра конструкций данных, как это определено в ИСО/МЭК 15961-2*;

15 — De-Compacted-Data (разуплотненные данные): это означает, что объект в ответе был разуплотнен с использованием правил ИСО/МЭК 15962 и восстановлен в исходный формат ввода приложения.

Как правило, следует применять уплотнение, поскольку оно повышает эффективность кодирования в радиочастотной метке. Объекты, уже закодированные по правилам кодирования в формате UTF-8, должны быть определены с помощью значения параметра уплотнения (Compact-Parameter) (2), чтобы впоследствии при считывании объекта было понятно, что он должен быть обработан устройством декодирования для формата UTF-8, чтобы получить окончательное представление для приложения, принимающего данные. Значение параметра уплотнения (Compact-Parameter) (0) должно использоваться только в том случае, если есть веская причина не применять уплотнение данных. Причинами для этого могут быть предварительное уплотнение в соответствии с правилами приложения или если объект был зашифрован. В обоих случаях, несмотря на то, что уплотнение возможно и процесс декодирования полностью восстановит объект, первоначально заданное значение параметра 0 или 2 будет потеряно, и принимающее приложение не сможет выполнить последующую обработку.

Альтернативой базовому уплотнению является кодирование набора объектов по правилам Packed Objects (упакованные объекты). Это можно сделать только в том случае, если администраторы приложений предпочтут принять схему путем создания таблицы индексов, ссылки на источники которой должны быть предоставлены в реестре конструкций данных. Затем для каждой радиочастотной метки, которая должна быть закодирована по правилам упакованных объектов, значение параметра уплотнения (Compact-Parameter) (3) применяется ко всему набору объектов, подлежащих кодированию.

Аналогично, если данные должны быть закодированы по правилам Tag Data Profiles (профили данных радиочастотных меток), значение параметра уплотнения (Compact-Parameter) (4) применяется ко всему набору объектов, подлежащих кодированию.

Во время процесса декодирования в зависимости от входных условий применяются следующие кодовые значения ответов:

- если аргумент команды имел значение Application-Defined (определено приложением) (0), то совместимый процессор данных (data processor) не выполнил уплотнение и применил соответствующий код уплотнения к кодированию в радиочастотной метке. Поскольку устройство декодирования не может интерпретировать закодированные байты, ответом является Application-Defined (определено приложением) (0);

* *См.* [*9*].

- если аргумент команды имел значение Compact (уплотнение) (1), то совместимый процессор данных (data processor) применил надлежащий код уплотнения к кодированию в радиочастотной метке. Поскольку устройство декодирования может интерпретировать закодированные байты, ответом является De-Compacted-Data (разуплотненные данные) (15);

- если аргументом команды были UTF8-Data (данные в формате UTF-8) (2), совместимый процессор данных (data processor) не выполнил уплотнение и применил соответствующий код уплотнения к кодированию в радиочастотной метке. Поскольку устройство декодирования не может интерпретировать закодированные байты, ответом является UTF8-Data (данные в формате UTF-8) (2);

- если аргументом команды был Pack-Objects (упакованные объекты) (3), то совместимый процессор данных (data processor) выполнил кодирование по правилам, заданным для объекта, в соответствии с указанной таблицей идентификаторов (ID Table). Устройство декодирования использует одну и ту же таблицу идентификаторов для интерпретации закодированного байта и возвращает каждый объект данных вместе с ответом De-Compacted-Data (разуплотненные данные) (15);

- если аргументом команды был Tag-data-Profile (профиль данных радиочастотных меток) (4), то совместимый процессор данных (Data Processor) выполнил кодирование в соответствии с правилами, заданными для объекта, как определено в указанной таблице идентификаторов. Устройство декодирования использует одну и ту же таблицу идентификаторов для интерпретации закодированного байта и возвращает каждый объект данных вместе с ответом De-Compacted-Data (разуплотненные данные) (15);

- если аргументом команды был Monomorphic-UII (мономорфный идентификатор UII) (5), то совместимый процессор данных (Data Processor) выполнил кодирование по правилам, заданным для объекта, как определено идентификатором AFI и регистром конструкций данных (Data Constructs register). Устройство декодирования при распознавании идентификатора AFI использует правило уплотнения, объявленное в регистре конструкций данных, и возвращает данные объекта с ответом De-Compacted-Monomorphic-UII (разуплотненный мономорфный идентификатор UII) (14). Он также возвращает идентификатор объекта (Object-Identifier), как определено регистром конструкций данных.

Примечание — Несмотря на то, что устройство декодирования необходимо для поиска правил уплотнения и разуплотнения, как было объявлено с помощью идентификатора AFI из регистра конструкций данных, ответы от данных считывания не включают идентификатор AFI. Этот аргумент ответа необходим для обеспечения возможности выполнения дополнительных процедур сопоставления.

7.3.7 Блокировка объекта (Object-Lock)

Аргумент команды Object-Lock (блокировка объекта) требует, чтобы процессор данных (data processor) построил кодированный набор данных [закодированный идентификатор объекта (Object-Identifier) или относительный идентификатор объекта (Relative-OID), объект (Object), прекурсор (Precursor) и другие связанные синтаксические компоненты] таким образом, что все связанные байты были сохранены и заблокированы способом блочного выравнивания на карте логической памяти. Процес блокировки подразумевает, что байты, обработанные таким образом, будут постоянно закодированы в радиочастотной метке или могут быть разблокированы только в некоторых типах радиочастотных меток с использованием соответствующих паролей.

7.4 Другие аргументы команды

7.4.1 Пароль доступа (Access-Password)

Наличие аргумента Access-Password (пароль доступа) требует, чтобы процессор данных (data processor) передал его устройству опроса с тем, чтобы проверить аргумент Access-Password (пароль доступа) в команде на соответствие с паролем доступа в радиочастотной метке. Результатом сравнения является разрешение дополнительных действий с радиочастотной меткой. Несоответствие приводит к тому, что радиоинтерфейс отклоняет команду.

7.4.2 Дополнительные прикладные биты (Additional-App-Bits)

Аргумент команды Additional-App-Bits (дополнительные прикладные биты) используется для расширения критериев поиска данных идентификатора объекта (Object-Identifier), закодированных в памяти идентификатора UII радиочастотной метки с сегментированной памятью. Аргумент Additional-App-Bits (дополнительные прикладные биты) добавляют в качестве суффикса к идентификаторам AFI и DSFID.

7.4.3 Блокирование идентификатора AFI (AFI-Lock)

Аргумент команды AFI-Lock (блокирование идентификатора AFI) используется для определения, подлежит ли блокированию идентификатор AFI или нет. Это аргумент типа данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ). Если значение аргумента — TRUE (ИСТИНА), устройство опроса должно заблокировать идентификатор AFI, чтобы гарантировать, что конкретная радиочастотная метка может

использоваться только в соответствии с предписаниями. Процесс блокировки должен приводить к тому, что обрабатываемые таким способом байты кодируются в радиочастотной метке перманентно или могут быть разблокированы только в некоторых типах радиочастотных меток с использованием соответствующих паролей.

Примечание — В некоторых типах радиочастотных меток идентификатор AFI является частью непрерывной строки с кодированными наборами данных. В таких случаях аргумент AFI-Lock (блокирование идентификатора AFI) не может применяться.

7.4.4 Добавление к существущей блочной записи (Append-To-Existing-Multiple-Record)

Аргумент команды Append-To-Existing-Multiple-Record (добавление к существущей блочной записи) является аргументом типа данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ). Если его значение TRUE (ИСТИНА), то объекты данных должны быть добавлены к существующей блочной записи при условии, что процессор данных (data processor) проведет различные проверки. Если его значение FALSE (ЛОЖЬ), то создается новая блочная запись.

7.4.5 Емкость записи, определяемая приложением (Application-Defined-Record-Capacity)

Аргумент команды Application-Defined-Record-Capacity (емкость записи, определяемая приложением) является аргументом типа данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ), при значении FALSE (ЛОЖЬ) процессор данных (data processor) автоматически определяет, что емкость для данной блочной записи просто основана на кодированных данных, а затем увеличивается до выравнивания по блоку. Если установлено значение TRUE (ИСТИНА), то приложение должно установить размер памяти, назначенный для записи, с использованием аргумента Record-Memory-Capacity (емкость памяти для записи).

7.4.6 Предотвращение дублирования (Avoid-Duplicate)

Аргумент команды Avoid-Duplicate (предотвращение дублирования) используется, чтобы гарантировать, что идентификатор объекта (Object-Identifier) еще не закодирован на карте логической памяти. Это аргумент типа данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ). Если установлено значение TRUE (ИСТИНА), устройство опроса проверяет все идентификаторы объектов на карте логической памяти. Если идентификатор объекта уже существует, команда записи прерывается и в коде завершения (Completion-Code) возвращает значение Duplicate-Object (дублирование объекта) (10). Если установлено значение FALSE (ЛОЖЬ), устройство опроса должно записать объект без какой-либо проверки.

7.4.7 Индикатор наличия батареи (Battery-Assist-Indicator)

Аргумент команды Battery-Assist-Indicator (индикатор наличия батареи) используется приложением, когда радиочастотная метка поддерживает функцию индикации состояния батареи. Аргумент обычно используется, когда протокол радиоинтерфейса не имеет функции для поддержки такой индикации.

7.4.8 Выравнивание блока (Block-Align)

Аргумент команды Block-Align (выравнивание блока) используется для определения того, необходимо ли записанные в радиочастотной метке объекты выравнивать по началу блока. Это аргумент типа данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ). Если установлено значение TRUE (ИСТИНА), устройство опроса должно выровнять каждый объект, записанный в радиочастотной метке, по границе блока, как определено конкретной архитектурой радиочастотной метки или ее изготовителем.

7.4.9 Выравнивание упакованных объектов по блоку (Block-Align-Packed-Object)

Аргумент команды Block-Align-Packed-Object (выравнивание упакованных объектов по блоку) используется для определения того, должны ли упакованные объекты, записанные в радиочастотной метке, выравниваться по началу блока. Это аргумент типа данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ). Если установлено значение TRUE (ИСТИНА), устройство опроса должно выровнять каждый упакованный объект, записанный в радиочастотной метке, по границе блока, как определено архитектурой конкретной радиочастотной метки или ее изготовителем.

7.4.10 Проверка дубликата (Check-Duplicate)

Аргумент команды Check-Duplicate (проверка дубликата) используется для вызова особого процесса, определенного командой (например, считывания или удаления). Это аргумент логического типа.

Если установлено значение TRUE (ИСТИНА), устройство опроса проверяет все идентификаторы объектов (Object-Identifiers) в карте логической памяти. Команда должна быть завершена только в отсутствие дубликата. В противном случае в качестве кода завершения (Completion-Code) возвращается значение DuplicateObject (дубликат объекта) (10).

Если для аргумента команды Check-Duplicate (проверка дубликата) установлено значение FALSE (ЛОЖЬ), устройство опроса должно реагировать на первое появление идентификатора объекта, ассоциированного объекта и прекурсора. В этом случае могут присутствовать дубликат идентификатора объекта и ассоциированный объект (возможно, с другим значением), а также прекурсор.

7.4.11 Индикатор CRC данных (Data-CRC-Indicator)

Аргумент команды Data-CRC-Indicator (индикатор CRC данных) в команде дает указание процессору данных (data processor) на добавление кода CRC-16 либо к каждому отдельному набору данных, либо ко всем закодированным данным, или и к тем, и другим. Аргумент Data-CRC-Indicator (индикатор CRC данных) в ответном сообщении указывает, что процессор данных (data processor) подтвердил код CRC-16, выделил его из ответа и передал интерпретируемый идентификатор объекта (Object-Identifier) и связанный с ним объект (Object) приложению.

7.4.12 Длина записи данных (Data-Length-Of-Record)

Этот аргумент ответа предоставляет размер закодированной блочной записи (Multiple Record) с точки зрения размера блока записи, как это закодировано в EBV-8 в преамбуле записи.

7.4.13 Метод удаления блочной записи (Delete-MR-Method)

Аргумент команды Delete-MR-Method (метод удаления блочной записи) используется приложением для вызова одного из двух методов для устройства опроса, чтобы удалить блочную запись. Этот аргумент команды является целочисленной переменной, принимающей следующие кодовые значения:

0 — отметить запись как удаленную;

1 — физически удалить запись.

Детали процесса определены в 10.28.1.

7.4.14 Индикатор EBV-8 длины каталога (Directory-Length-EBV8-Indicator)

Аргумент команды Directory-Length-EBV8-Indicator (индикатор EBV-8 длины каталога*) используется приложением для подтверждения того, что для процессора данных (data processor) имеется достаточное пространство в заголовке блочной записи (MR-header) для кодирования фактической длины каталога. Этот аргумент команды является целочисленной переменной, принимающей следующие кодовые значения:

1 — однобайтовый формат EBV-8, который может поддерживать размер каталога до 127 блоков;

2 — двухбайтовый формат EBV-8, который следует использовать, когда логическая память имеет большой размер и/или ожидается, что в ней будет закодировано большое число записей. При этом поддерживается размер каталога до 16 383 блоков.

Значение 2 используется для каталога, который может быть изначально небольшим по размеру, однако ожидается, что в будущем его размер будет увеличен.

7.4.15 Блокировка идентификатора DSFID (DSFID-Lock)

Аргумент команды DSFID-Lock (блокировка идентификатора DSFID) (ранее — команда storageFormatLock) используется для определения того, должна ли быть заблокирована функция идентификатора DSFID. Это аргумент типа данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ). Если установлено значение TRUE (ИСТИНА), устройство опроса должно заблокировать идентификатор DSFID, чтобы гарантировать, что конкретная радиочастотная метка может использоваться только в предписанном порядке. Процесс блокировки должен приводить к тому, что обрабатываемые таким способом байты будут постоянно закодированы неизменно в памяти радиочастотной метки или могут быть разблокированы только в некоторых типах радиочастотных меток с использованием соответствующих паролей.

Примечание — В некоторых типах радиочастотных меток идентификатор DSFID является частью непрерывной строки с закодированными наборами данных. В таких случаях он не может быть заблокирован независимо.

7.4.16 Байты-заполнители идентификатора DSFID (DSFID-Pad-Bytes)

Аргумент команды DSFID-Pad-Bytes (байты-заполнители идентификатора DSFID) указывает число байтов, которые приложение намеривается предоставить для использования в расширенном идентификаторе DSFID (Extended-DSFID), в дополнение к тем, что создаются процессором данных (data processor). Например, это может служить для предоставления дополнительного пространства для кодирования данных, которые будут добавлены и обновлены в будущем. Когда он включен в ответ, дополнительные байты-заполнители могут быть добавлены процессором данных (data processor) для выравнивания по границе блокирующего блока, если расширенный идентификатор DSFID (Extended-DSFID) должен быть заблокирован или если первый набор данных (Data-Set) должен быть заблокирован.

7.4.17 Размер редактируемого указателя (Editable-Pointer-Size)

Аргумент команды Editable-Pointer-Size (размер редактируемого указателя) используется для указания того, что упакованный объект, созданный явно или неявно посредством команды Write-Objects (записать объекты), должен допускать последующие изменения его содержимого или структуры. Этот аргумент команды представлен как ненулевое целочисленное значение, указывающее, что созданный

* *EBV — сокращение от Extensible bit vector.*

упакованный объект (Packed Object) должен быть доступен для редактирования путем добавления дополнительного подраздела к окончанию раздела Object Info section (информация об объекте) упакованного объекта. Указателем(-ями) дополнений упакованного объекта (Addendum Packed-Object) должно быть число битов, приведенных в этом аргументе команды. Значение по умолчанию для этого аргумента равно нулю, что указывает на то, что подсекции дополнений (addendum subsection) отсутствуют, и поэтому упакованный объект не редактируется.

7.4.18 Емкость кодированной памяти (Encoded-Memory-Capacity)

Этот аргумент ответа предоставляет размер, зарезервированный для блочной записи, с точки зрения размера блока записи, как это было закодировано в EBV-8 в преамбуле записи.

7.4.19 Код EPC (EPC-Code)

Аргумент команды EPC-Code (код EPC) представляет собой любой из уникальных кодов, определенных стандартом TDS*. Структура каждого кодового значения EPC самодекларируется исходя из значения его 8-битового заголовка. Хотя некоторые кодовые значения не выравниваются по 8-битовой границе, их необходимо округлять до 8-битовых байтов для обработки процессором данных (data processor) и округлять до 16-битовых слов для кодирования в соответствующей радиочастотной метке.

7.4.20 Индикатор полнофункционального датчика (Full-Function-Sensor-Indicator)

Аргумент команды Full-Function-Sensor-Indicator (индикатор полнофункционального датчика) используется приложением, когда радиочастотная метка поддерживает полнофункциональный датчик. Аргумент обычно используется, когда протокол радиоинтерфейса не имеет функции для поддержки такой индикации.

7.4.21 Дуга иерархического идентификатора (Hierarchical-Identifier-Arc)

Аргумент ответа Hierarchical-Identifier-Arc (дуга иерархического идентификатора) представляет собой целочисленное значение, преобразованное из значения EBV-8, закодированное в преамбуле блочных записей из иерархической записи.

7.4.22 Родительский идентификатор (Identifier-Of-My-Parent)

Аргумент команды и ответа Identifier-Of-My-Parent (родительский идентификатор) используется для блочных записей, которые являются частью иерархии, и имеет значение иерархического кода этой записи.

7.4.23 Метод идентификации (Identify-Method)

Аргумент команды Identify-Method (метод идентификации) используется для определения того, должны ли идентифицироваться все или некоторые из радиочастотных меток, принадлежащих определенному выбранному идентификатору AFI в рабочей области. Этот аргумент команды представляется целочисленной переменной, принимающей следующие кодовые значения:

0 — Inventory-All-Tags (инвентаризация всех радиочастотных меток);

1 — Inventory-At-Least *(инвентаризация не менее указанного числа радиочастотных меток);

2 — Inventory-No-More-Than (инвентаризация не более указанного числа радиочастотных меток);

3 — Inventory-Exactly (инвентаризация указанного числа радиочастотных меток);

4—15 зарезервированы для будущего использования.

Для каждого значения аргумента, включая значение Inventory-All-Tags (инвентаризация всех радиочастотных меток), необходимо указать число считываемых радиочастотных меток с помощью аргумента Number-Of-Tags (число радиочастотных меток) (см. 7.4.43). Более точные рекомендации предоставляются в команде Inventory-Tags (инвентаризации радиочастотных меток) (см. 10.4).

7.4.24 Тип идентификатора (ID-Type)

Аргумент команды ID-Type (тип идентификатора) используется для указания того, что упакованный объект (Packed-Object), созданный явно или неявно командой Write-Objects (запись объектов), и должен быть создан как тип ID List (список идентификаторов) или ID Map (карта идентификаторов). Этот аргумент команды представляется целочисленной переменной, принимающей следующие кодовые значения:

0 — ID List (список идентификаторов);

1 — ID Map (карта идентификаторов);

2 — 15 зарезервированы для будущего использования.

7.4.25 Дуга экземпляра (Instance-Of-Arc)

Аргумент ответа Instance-Of-Arc (дуга экземпляра) представляет собой целочисленное значение, преобразованное из значения EBV-8, закодированного в преамбуле блочной записи.

---

* *См.* [*22*].

7.4.26 Пароль на уничтожение (Kill-Password)

Аргумент команды Kill-Password (пароль на уничтожение) требует, чтобы процессор данных (data processor) передал его устройству опроса, при этом аргумент Kill-Password (пароль на уничтожение) в команде должен совпадать с паролем, записанным в радиочастотной метке. Результатом совпадения является то, что радиочастотная метка перестает функционировать. Несоответствие приводит к тому, что радиоинтерфейс отклоняет команду.

7.4.27 Длина маски (Length-Of-Mask)

Аргумент команды Length-Of-Mask (длина маски) используется вместе с аргументами команды Pointer (указатель адреса) и Tag-Mask (маска радиочастотной метки) для определения критериев поиска данных в стандарте TDS*, закодированных в памяти идентификатора UII радиочастотной метки с сегментированной памятью.

7.4.28 Блокировка позиции каталога (Lock-Directory-Entry)

Аргумент команды Lock-Directory-Entry (блокировка позиции каталога) соответствует типу данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ), и, если установлено значение TRUE (ИСТИНА), устройство опроса должно заблокировать входпозицию каталога для блочной записи.

7.4.29 Блокировка заголовка блочных записей (Lock-Multiple-Records-Header)

Аргумент команды Lock-Multiple-Records-Header (блокировка заголовка блочных записей) используется для определения факта: заблокированы ли все, некоторые или ни один из заголовков блочных записей. Этот аргумент команды представляется целочисленной переменной, принимающей следующие кодовые значения:

0 — не заблокирован;

1 — полностью заблокирован;

2 — поле Number-of-records (число записей) остается разблокированным, предыдущие поля заблокированы;

3 — длина данных в поле Directory (каталог) остается разблокированной, все остальные поля, включая поле Number-of-records (число записей), заблокированы;

4 — длина данных в поле Directory (каталог) и поле Number-of-records (число записей) остаются разблокированными, все остальные поля заблокированы.

7.4.30 Блокировка преамбулы записи (Lock-Record-Preamble)

Аргумент команды Lock-Record-Preamble (блокировка преамбулы записи) соответствует типу данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ), и, если установлено значение TRUE (ИСТИНА), устройство опроса должно заблокировать поля в преамбуле.

7.4.31 Аргументы блокировки сегмента идентификатора UII (Lock-UII-Segment-Arguments)

Аргумент команды Lock-UII-Segment-Arguments (аргументы блокировки сегмента идентификатора UII) используется для определения того, какие составляющие части сегмента идентификатора UII радиочастотной метки типа D должны быть заблокированы. Данный аргумент имеет приоритет над аргументом блокировки также и в связанной команде.

7.4.32 Максимальная длина для приложения (Max-App-Length)

Аргумент команды Max-App-Length (максимальная длина для приложения) используется для определения максимальной длины уплотненного закодированного набора данных. Он используется в командах, которым необходимо ограничить команды считывания радиоинтерфейса для более быстрых транзакций, например для считывания идентификатора UII, закодированного в первом наборе данных в радиочастотной метке.

7.4.33 Банк памяти (Memory-Bank)

Аргумент команды Memory-Bank (банк памяти) используется для определения того, какая часть сегментированной памяти радиочастотной метки используется для кодирования данных. После этого процессор данных (data processor) вызывает правила, специфичные для банка памяти, для достижения надлежащего кодирования радиочастотной метки.

7.4.34 Блокировка банка памяти (Memory-Bank-Lock)

Аргумент команды Memory-Bank-Lock (блокировка банка памяти) используется для определения того, какая часть сегментированной памяти радиочастотной метки должна быть заблокирована.

7.4.35 Кодирование объема памяти (Memory-Length-Encoding)

Аргумент команды Memory-Length-Encoding (кодирование объема памяти) дает указание процессору данных (data processor) на кодирование либо объема памяти, либо длины закодированных

* *См.* [22].

данных, либо и того, и другого, выраженных числом блоков. Аргумент Memory-Length-Encoding (кодирование объема памяти) в ответе указывает, что процессор данных (data processor) рассчитал соответствующую длину и закодировал это как часть расширенного идентификатора DSFID.

7.4.36 Сегмент памяти (Memory-Segment)

Аргумент команды Memory-Segment (сегмент памяти) используется для определения того, в какой части сегментированной памяти радиочастотной метки должны быть закодированы данные. Это похоже на функцию аргумента Memory-Bank (банк памяти) (см. 7.4.33), за исключением того, что радиочастотная метка, которой этот аргумент соответствует, может адресовать несколько сегментов в одной и той же транзакции радиоинтерфейса.

7.4.37 Тип памяти (Memory-Type)

Аргумент команды Memory-Type (тип памяти) используется для определения структуры памяти для кодирования мономорфного идентификатора UII (Monomorphic-UII), эффективно инструктируя процессор данных (data processor), какой из необязательных и условных аргументов и процессов необходимо применить в процессе кодирования.

7.4.38 Длина каталога блочных записей (Multiple-Records-Directory-Length)

Аргумент ответа Multiple-Records-Directory-Length (длина каталога блочных записей) имеет значение EBV-8.

7.4.39 Индикатор характеристик блочных записей (Multiple-Records-Features-Indicator)

Аргумент команды и ответа Multiple-Records-Features-Indicator (индикатор характеристик блочных записей) — это карта битов со следующей структурой:

- бит 8 со значением 1_2 указывает, что все записи используют один и тот же метод доступа. Если этот бит = 0_2, то биты с 7 по 4 = 0000_2;
- биты с 7 по 4 определяют метод доступа, если он применяется последовательно. Если бит 8 = 0_2, допустима строка для битов с 7 по 4:
  - 0000_2 указывает, что все записи используют метод доступа No-Directory (без каталога),
  - 0001_2 — не разрешено,
  - 0010_2 указывает, что все записи используют Packed-Objects Access-Method (метод доступа к упакованным объектам),
  - 0011_2 указывает, что все записи используют Tag-Data-Profile Access-Method (метод доступа с профилем данных радиочастотных меток),
  - 0100_2 — не разрешено,
  - диапазон от 0101_2 до 1111_2 зарезервирован для дополнительных методов доступа (Access-Method);
- бит 3 со значением 1_2 указывает, что некоторые из записей находятся в иерархической связи с другими записями;
- бит 2 определяет, полностью ли поддерживается поле number-of-records (число записей) (например, обновляется каждый раз при добавлении новой записи), или корректно ли текущее значение поля number-of-records (число записей). Значение 1 указывает на то, что значение поля number-of-records (число записей) корректно, значение 0_2 указывает на то, что поле number-of-records (число записей) может быть некорректным;
- бит 1 со значением 1_2 указывает на то, что имеется дополнительный байт для индикации дополнительных функций. В настоящее время он зарезервирован для будущего использования (RFU) и поэтому установлен в 0_2.

7.4.40 Биты NSI (NSI-Bits)

Аргумент команды NSI-Bits* (биты NSI) определяет 9-разрядный код в соответствии со стандартом TDS**, используемый в качестве префикса для кодового значения EPC при кодировании в соответствующей радиочастотной метке.

7.4.41 Число экземпляров в списке элементов данных (Number-In-Data-Element-List)

Аргумент команды Number-In-Data-Element-List (число экземпляров в списке элементов данных) используется только в иерархической записи, которая является списком элементов данных (data element list). Значение представляет собой число экземпляров элемента данных, подлежащих кодированию.

7.4.42 Число записей (Number-Of-Records)

Аргумент ответа Number-Of-Records (число записей) используется одним из двух способов. Если бит 2 аргумента Multiple-Records-Features-Indicator (индикатор характеристик блочных записей) равен:

---

* *NSI — Numbering System Identifier (идентификатор системы нумерации).*

** *См.* [22].

0_2 — то это поле не поддерживается, а значение аргумента Number-Of-Records (число записей) должно быть нулевым, однако любое другое значение должно игнорироваться, например, если некоторое число записей было первоначально сохранено, но позже было решено остановить эту функцию;

1_2 — то это поле полностью поддерживается по мере добавления новых записей. Нулевое значение для аргумента Number-Of-Records (число записей) кодируется при создании заголовка блочных записей.

7.4.43 Число радиочастотных меток (Number-Of-Tags)

Аргумент команды Number-Of-Tags (число радиочастотных меток) используется для определения ограничений к установленному аргументу Identify-Method (метод идентификации). Это целочисленное значение в диапазоне от 0 до 65535.

7.4.44 Множитель смещения объектов (Objects-Offsets-Multiplier)

Аргумент команды Objects-Offsets-Multiplier (множитель смещения объектов) определяет размер памяти в битах, которые запрашиваются для хранения смещений объекта, когда параметр Directory-Type (тип каталога) в аргументе Packed-Objects-Constructs (конструкции упакованного объекта) является Packed-Object offset (смещением упакованного объекта). Реализация должна использовать параметр для надлежащей калибровки структуры AuxMap structure в упакованном объекте (Packed-Object).

7.4.45 Тип каталога упакованного объекта (Packed-Object-Directory-Type)

Аргумент команды Packed-Object-Directory-Type (тип каталога упакованного объекта) определяет, является ли упакованный объект каталогом (directory), и в этом случае каталог (directory) упакованного объекта какого типа был создан. Каталог присутствие/отсутствие (Presence/Absence directory) предоставляет битовую карту значений идентификаторов, закодированных на любом из упакованных объектов в радиочастотной метке, но не указывает, где кодируются данные. Каталог с индексными полями (index field directory) добавляет порядковое значение упакованного объекта, содержащее конкретное значение идентификатора. Например, поле 3-битового индекса определяет, какой из восьми упакованных объектов будет доступен. Каталог смещений (offset directory) предоставляет начальный адрес упакованного объекта, содержащий определенное значение идентификатора.

Пустой каталог (null directory) (т. е. предоставление пространства для каталога) создается путем указания в этом аргументе положения указателя, установив ненулевое значение для аргумента PO-Directory-Size (размер каталога упакованного объекта).

Если упакованный объект не является каталогом (directory), то для этого аргумента устанавливается нулевое значение.

7.4.46 Пароль (Password)

Аргумент команды Password (пароль) определяет строку байтов, которая представляет собой кодовое значение, определяемое аргументом Password-Type (тип пароля), в команде, которая требуется для сопоставления с аналогично определенным кодовым значением в радиочастотной метке. Соответствие дает разрешения для выполнения дополнительных действий с радиочастотной меткой. Несоответствие приводит к отклонению любой связанной команды.

7.4.47 Тип пароля (Password-Type)

Аргумент команды Password-Type (тип пароля) определяет значение аргумента Password (пароль), используемого для предоставления разрешений для выполнения дополнительных действий с радиочастотной меткой.

7.4.48 Размер каталога упакованного объекта (PO-Directory-Size)

Аргумент команды PO-Directory-Size (размер каталога упакованного объекта) определяет размер указателя пустого каталога [(null) directory pointer], созданного с помощью упакованного объекта. Этот аргумент команды представляется как ненулевое целочисленное значение, которое должно использоваться для размера указателя пустого каталога [(null) directory pointer], закодированного в упакованном объекте.

7.4.49 Длина индекса упакованного объекта (PO-Index-Length)

Аргумент команды PO-Index-Length (длина индекса упакованного объекта) [используется для указания размера PO-Index-Length (длины индекса упакованного объекта)] для структуры AuxMap structure, которая будет создана, если параметр Packed-Object-Directory-Type (тип каталога упакованного объекта) — это поле индекса (index field) упакованного объекта.

7.4.50 Указатель адреса (Pointer)

Аргумент команды Pointer (указатель адреса) используется в сочетании с аргументами команды Length-Of-Mask (длина маски) и Tag-Mask (маска радиочастотной метки) для определения критериев поиска данных в стандарте TDS*, закодированных в блоке памяти идентификатора UII радиочастотной метки с сегментированной памятью.

* *См.* [22].

7.4.51 Адресная ссылка на каталог блочных записей (Pointer-To-Multiple-Records-Directory)

Аргумент команды и ответа Pointer-To-Multiple-Records-Directory (адресная ссылка на каталог блочных записей) используется для определения самого высокого номера блока на начальном адресе каталога. Он определяется как значение в EBV-8. Если каталог не был изначально закодирован, то строка в EBV-8 одной и той же фиксированной длины, что и для начальной точки каталога, должна быть закодирована с нулевым значением. Каталог кодируется в обратной последовательности блоков, так что следующий блок каталога находится на следующем, более низком адресе. Начало каталога не обязательно является наивысшим адресом в логической памяти, поскольку для размещения по этому адресу могут быть указаны другие аппаратные функции радиочастотной метки. Чтобы определить начальный адрес каталога, необходимо будет вызвать команды радиоинтерфейса, которые определяют распределение памяти (memory mapping).

7.4.52 Тип считываемой записи (Read-Record-Type)

Аргумент команды Read-Record-Type (тип считываемой записи) используется для идентификации различных логических структур из радиочастотной метки. Этот аргумент команды имеет целочисленное значение. Применяются следующие кодовые значения:

0 — Read-Multiple-Records-Header (считывание заголовка блочных записей);

1 — Read-Multiple-Records-Header-Plus-1st-Preamble (считывание заголовка блочных записей и первой преамбулы);

2 — Read-Multiple-Records-Directory (считывание каталога блочных записей);

3 — Read-Preamble-Specific-Multiple-Record (считывание преамбулы заданной блочной записи);

4 — Read-All-Record-OIDs-Specific-Record-Type (считывание всех записей идентификаторов объектов с заданным типом записи);

5 — Read-OIDs-Specific-Multiple-Record (считывание идентификаторов объектов заданной блочной записи);

6 — Read-All-Objects-Specific-Multiple-Record (считывание всех объектов с заданной блочной записью);

7 — Read-Multiple-Objects-Specific-Multiple-Record (считывание блочных объектов, заданных блочной записью);

8 — Read-1st-Objects-Specific-Multiple-Record (считывание заданной блочной записи первых объектов);

9 — Read-Data-Element-List-Specific-Multiple-Record (считывание заданной блочной записи списка элементов данных).

Если выбрано значение Read-Multiple-Records-Header (считывание заголовка блочной записи), процессор данных (data processor) возвращает интерпретацию MR-header (заголовок блочной записи) в аргументе Multiple-Records-Header-Structure (структура заголовка блочных записей).

Если выбрано значение Read-Multiple-Records-Header-Plus-1st-Preamble (считывание заголовка блочных записей и первой преамбулы), процессор данных (data processor) возвращает интерпретацию MR-header (заголовок блочной записи) в аргументе Multiple-Records-Header-Structure (структура заголовка файла блочной записи) и интерпретацию преамбулы в первой записи в аргументе Multiple-Records-Preamble-Structure (структура преамбулы блочных записей).

Если выбрано значение Read-Multiple-Records-Directory (считывание каталога блочных записей), процессор данных (data processor) возвращает интерпретацию каталога с блочными записями в аргументе Multiple-Records-Directory-Structure (структура каталога блочных записей).

Для кодовых значений Read-Record-Type (тип считываемой записи) от 3 до 9 требуется использование одного или нескольких идентификаторов объектов (Object-Identifiers) в команде для процессора данных (data processor) для вызова соответствующих процессов. Три очень специфических формата идентификаторов объекта применяются к блочным записям:

- для блочных записей, которые не являются частью иерархии, структура следующая: 1.0.15961.401.{Data-Format}.{sector identifier}.{record type}.{instance-of}.{Relative-OID of data element} (1.0.15961.401.{формат данных}.{идентификатор сектора}.{ }.{экземпляр}.{относительный идентификатор объекта элемента данных});

- для блочных записей, которые являются частью иерархии, но не списка элементов данных (data element list), структура следующая: 1.0.15961.402.{Data-Format}.{sector identifier}.{record type}.{hierarchical id}.{Relative-OID of element} (1.0.15961.402.{формат данных}.{идентификатор сектора}.{тип записи}.{иерархический идентификатор}.{относительный идентификатор объекта элемента данных});

- для блочной записи, которая является списком элементов данных (data element list), структура следующая: 1.0.15961.403.{Data-Format}.{sector identifier}.{record type}.{hierarchical id}.{data element} (1.0.15961.403.{формат данных}.{идентификатор сектора}.{тип записи}.{иерархический идентификатор}.{элемент данных}).

Примечание — Этот идентификатор объекта требует, чтобы ответ включал все номера элементов списка.

Первые две структуры идентификатора объекта применяются к кодовым значениям от 3 до 8 Read-Record-Type (тип считываемой записи). Третья структура идентификатора объекта из вышеперечисленных применяется только к кодовым значениям 3, 4 и 9 Read-Record-Type (тип считываемой записи).

Если выбрано значение Read-Preamble-Specific-Multiple-Record (считывание преамбулы заданной блочной записи), команда должна включать в себя один идентификатор объекта в списке считанных объектов, который безусловно определен вплоть до дуги типа записи (record type arc), а также дуги экземпляра (instance-of arc) (если применимо) или дуги иерархического идентификатора (hierarchical id arc) (если применимо). Процессор данных (data processor) возвращает интерпретацию преамбулы записи в аргументе Multiple-Records-Preamble-Structure (структура преамбулы блочных записей).

Выбор значения Read-All-Record-OIDs-Specific-Record-Type (считывание всех записей идентификаторов объектов с заданным типом записи) полезен для идентификации серий записей истории одного и того же типа или набора записей в иерархии одного и того же типа. В этом случае команда должна включать единственный идентификатор объекта в списке считанных объектов, который определяется только до дуги типа записи. Процессор данных (data processor) возвращает список идентификаторов объектов на один уровень ниже в Read-OIDs-Response-List (список ответов на считывание идентификаторов объектов), т. е. либо с набором дуг экземпляров, либо с набором дуг иерархических идентификаторов.

Если выбрано значение Read-OIDs-Specific-Multiple-Record (считывание идентификаторов объектов заданной блочной записи), команда должна включать в себя один идентификатор объекта в списке считанных объектов, который однозначно определяется до дуги типа записи, а также дуги экземпляра (если применимо) или дуги иерархического идентификатора (если применимо). Процессор данных (data processor) возвращает список идентификаторов объектов, закодированных внутри записи в Read-OIDs-Response-List (список ответов на считывание идентификаторов объектов). Это не относится к спискам элементов данных (Data Element lists).

Если выбрано значение Read-All-Objects-Specific-Multiple-Record (считывание всех объектов с заданной блочной записью), команда должна включать в себя один идентификатор объекта в списке считанных объектов, который однозначно определен до дуги типа записи, а также дуги экземпляра (если применимо) или дуги иерархического идентификатора (если применимо). Процессор данных (data processor) возвращает список идентификаторов объектов и объектов, закодированных внутри записи в Read-Objects-Response-List (список ответов на считывание объектов). Это не относится к спискам элементов данных.

Если выбрано значение Read-Multiple-Objects-Specific-Multiple-Record (считывание блочных объектов с заданной блочной записью), команда должна включать в себя назначенные идентификаторы объектов, которые определены вплоть до конкретного элемента данных в списке считанных данных. Процессор данных (data processor) возвращает список назначенных идентификаторов объектов и объектов, закодированных внутри записи в Read-Objects-Response-List (список ответов на считывание объектов). Это не относится к спискам элементов данных.

Если выбрано значение Read-1st-Objects-Specific-Multiple-Record (считывание заданной блочной записи первых объектов), команда должна включать назначенный(ые) (nominated) идентификатор(ы) объекта(ов), который определен(ы) вплоть до конкретного(ых) элемента(ов) данных в Read-Objects List (список считанных объектов). Процессор данных (data processor) возвращает список назначенных (nominated) идентификаторов объектов и объектов, закодированных до максимальной длины для приложения (Max-App-Length), в пределах записи в Read-Objects-Response-List (список ответов на считывание объектов). Это не относится к спискам элементов данных.

Если выбрано значение Read-Data-Element-List-Specific-Multiple-Record (считывание заданной блочной записи списка элементов данных), команда должна включать в себя один идентификатор объекта в списке считанных объектов, который определяется с помощью элемента данных. Сначала процессор данных (data processor) проверяет, что запись является списком элементов данных (Data Element list). Если это так, он использует правила метода доступа для реконструкции идентификатора объекта до номера элемента списка, закодированного в списке элементов данных, и возвращает этот и связанные объекты, закодированные внутри записи, в Read-Objects-Response-List (список ответов на считывание объектов).

7.4.53 Тип считывания (Read-Type)

Аргумент команды Read-Type (тип считывания) используется для идентификации числа и местоположения идентификаторов объектов в команде считывания. Этот аргумент команды представлен как целочисленное значение, принимающее следующие кодовые значения:

0 — Read-1st-Objects (считывание первых объектов);

1 — Read-Multiple-Objects (считывание блочных объектов);

2 — Read-All-Objects (считывание всех объектов);

3 — Read-Monomorphic-UI (считывание мономорфных идентификаторов UII);

4 — 15 — зарезервированы для будущего использования.

Если выбрано значение Read-1st-Objects (считывание первых объектов), то команда должна включать кодовое значение Max-App-Length (максимальная длина для приложения), что эквивалентно числу байтов, предназначенных для считывания в приложении. Аргумент структурирован скорее для считывания последовательности идентификаторов объектов, чем идентификатора объекта только в первой позиции, и поэтому отличается от указанного в отмененном стандарте ИСО/МЭК 15961:2004*. Если аргумент Read-Type (тип считывания) установлен в значение Read-Multiple-Objects (считывание блочных объектов), это может применяться к одному или нескольким идентификаторам объектов.

Если выбрано значение Read-Monomorphic-UII (считывание мономорфных идентификаторов UII), оно дает указание процессору данных (data processor) на выполнение дополнительных проверок с помощью регистра конструкций данных ИСО/МЭК (ISO/IEC Data constructs register), чтобы обеспечить выполнение по команде надлежащего процесса.

7.4.54 Емкость памяти для записи (Record-Memory-Capacity)

Аргумент команды Record-Memory-Capacity (емкость памяти для записи) используется в командах для сохранения блочной записи с целью указать количество назначаемой памяти (выраженное числом блоков записи), как правило, для добавления к записи дополнительных элементов данных. Он используется только в том случае, если приложение должно перезагрузить автоматическую калибровку процессора данных (data processor), если аргумент Application-Defined-Record-Capacity (емкость записи, определенная приложением) установлен в значение TRUE (ИСТИНА).

7.4.55 Дуга с типом записи (Record-Type-Arc)

Аргумент ответа Record-Type-Arc (дуга с типом записи) представляет собой целочисленное значение, преобразованное из значения в EBV-8**, закодированного в преамбуле блочных записей (Multiple Records preamble).

7.4.56 Классификация типа записи (Record-Type-Classification)

Аргумент команды и ответа Record-Type-Classification (классификация типа записи) — это двоичная строка, которая идентифицирует класс кодируемой записи. Применяются следующие кодовые значения:

000_2 — автономная запись (stand-alone record) со значением дуги экземпляра (instance-of arc) = 0;

001_2 — автономная запись со значением дуги экземпляра > 0;

010_2 — иерархическая запись, верхний уровень;

011_2 — иерархическая запись, имеет как родителя, так и потомка(ов);

100_2 — иерархическая запись, список элементов данных (data element list);

101_2 — иная иерархическая запись, без последующих потомков (никогда не применяется к списку элементов данных);

110_2 — не относится к этой команде [поскольку она связана с удаленными (deleted) записями];

111_2 — зарезервировано.

7.4.57 Идентификатор сектора (Sector-Identifier)

Аргумент команды и ответа Sector-Identifier (идентификатор сектора) используется в заголовке блочной записи (MR-header) либо для указания истинного идентификатора сектора для всех записей в логической памяти, либо для указания того, что истинное значение варьируется между записями и истинное значение кодируется в записи. Этот аргумент команды и ответа представлен как целочисленная переменная, принимающая следующие кодовые значения:

0 — истинный идентификатор сектора, кодируется в каждой записи и может варьироваться между записями;

1 — используется для закрытых системных приложений;

---

* *См.* [*8*].

** *В ISO/IEC 15961-1:2021 ошибочно указано EBY-8.*

2 — тип записи имеет значение, равное относительному идентификатору объекта (Relative-OID) первого элемента данных, закодированного в записи. Например, если первый элемент данных имеет относительный идентификатор объекта (Relative-OID) = 7 для кодового значения продукции, тогда тип записи (Record type) = 7;

> 2 — идентификатор сектора (Sector-Identifier), назначаемый сектору администраторами словаря данных для управления своими собственными типами размещенных записей.

Если все записи имеют одинаковый идентификатор сектора, ненулевое значение является частью структуры идентификатора объекта в команде для записи или считывания записи, но должно быть включено отдельно для команды Configure-MultipleRecords-Header (сконфигурировать заголовок блочных записей).

7.4.58 Индикатор простого датчика (Simple-Sensor-Indicator)

Аргумент команды Simple-Sensor-Indicator (индикатор простого датчика) используется приложением, когда радиочастотная метка поддерживает простой датчик. Аргумент обычно используется, когда протокол радиоинтерфейса не имеет функции для поддержки такой индикации.

7.4.59 Начальный адрес записи (Start-Address-Of-Record)

Этот аргумент ответа предоставляет адрес первого байта кодированной блочной записи (Multiple Record) в контексте размера блока записи (write block size), как закодировано в EBV-8 в каталоге блочных записей (multiple-records directory).

7.4.60 Таблица идентификаторов профиля данных радиочастотных меток (Tag-Data-Profile-ID-Table)

Аргумент команды Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток) используется для идентификации конкретного протокола данных радиочастотной метки, который содержит правила уплотнения и форматирования. Для кодирования любых данных для данного приложения с использованием этого метода доступа необходимы особые правила.

7.4.61 Маска радиочастотной метки (Tag-Mask)

Аргумент команды Tag-Mask (маска радиочастотной метки) используется вместе с аргументами команды Length-Of-Mask (длина маски) и Pointer (указатель адреса) для определения критериев поиска данных в стандарте TDS*, закодированных в идентификаторе UII сегментированной памяти радиочастотной метки.

7.4.62 Обновление каталога блочных записей (Update-Multiple-Records-Directory)

Аргумент команды Update-Multiple-Records-Directory (обновление каталога блочных записей) является типом данных BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ), но обработка в логической памяти процессором данных (data processor) должна учитывать, существует ли уже каталог. Применяются следующие состояния и процессы:

- если каталог уже существует и для аргумента команды установлено значение TRUE (ИСТИНА), каталог полностью обновляется, включая любые ранее пропущенные позиции каталога;
- если каталог существовал ранее, а для аргумента команды установлено значение FALSE (ЛОЖЬ), аргумент игнорируется, и каталог полностью обновляется, включая любые ранее пропущенные позиции каталога;
- если каталог не существует и для аргумента команды установлено значение TRUE (ИСТИНА), каталог создается и полностью обновляется, включая любые ранее пропущенные позиции каталога;
- если каталог не существует и для аргумента команды установлено значение FALSE (ЛОЖЬ), каталог не создается.

7.4.63 Число слов (Word-Count)

Аргумент команды Word-Count (число слов) используется в сочетании с аргументом Word-Pointer (указатель слова) для определения числа закодированных байтов для считывания из радиочастотной метки с сегментированной памятью без какой-либо обработки, которую должен выполнять процессор данных (data processor).

7.4.64 Указатель слова (Word-Pointer)

Аргумент команды Word-Pointer (указатель слова) используется в сочетании с аргументом Word-Count (число слов) и определяет начальную ячейку памяти сегментированной памяти радиочастотной метки, начиная с которой должна считываться закодированная строка байтов.

---

* *См.* [22].

7.5 Имена полей, связанных с командами

7.5.1 Общие положения

В дополнение к аргументам команд (7.4) в командах и ответах используются следующие имена полей.

7.5.2 Набор данных (Data-Set)

Поле с именем Data-Set (набор данных) относится к непрерывной строке байтов начиная с прекурсора (Precursor), идентификатора объекта и до конечного байта уплотненного объекта в радиочастотной метке.

7.5.3 Ключевые идентификаторы (Identities)

Поле с именем Identities (ключевые идентификаторы) — это поле команды-ответа, которое представляет список идентифицированных идентификаторов Singulation-Ids.

7.5.4 Байт блокированной длины (Length-Lock Byte)

Поле с именем Length-Lock Byte (байт блокированной длины) — это поле команды-ответа, которое идентифицирует длину закодированных данных в сегменте, относящемся к предмету, а также предоставляет информацию о статусе блокировки страниц (lock status of pages) в радиочастотной метке.

7.5.5 Длина закодированных данных (Length-Of-Encoded-Data)

Поле с именем Length-Of-Encoded-Data (длина закодированных данных) — это поле команды-ответа, которое является частью расширенного идентификатора DSFID и идентифицирует длину закодированных данных в блоках.

7.5.6 Статус блокировки (Lock-Status)

Поле с именем Lock-Status (статус блокировки) — это поле команды-ответа, которое идентифицирует, заблокирован или не заблокирован закодированный пакет [например, набор данных (Data-Set)].

7.5.7 Карта логической памяти (Logical-Memory-Map)

Поле с именем Logical-Memory-Map (карта логической памяти) — это поле команды-ответа, которое представляет собой полную, но не декодированную строку байтов, которая была считана из радиочастотной метки.

7.5.8 Емкость памяти (Memory-Capacity)

Поле с именем Memory-Capacity (емкость памяти) — это поле команды-ответа, которое является частью расширенного идентификаторв DSFID и определяет длину закодированных данных в блоках.

7.5.9 Модуль идентификатора объекта (Module-OID)

Поле с именем Module-OID (модуль идентификатора объекта) идентифицирует отдельный модуль команды или ответа с полным идентификатором объекта, как определено в 7.3.3.

7.5.10 Число обнаруженных радиочастотных меток (Number-Of-Tags-Found)

Поле с именем Number-Of-Tags-Found (число обнаруженных радиочастотных меток) — это поле команды-ответа, которое возвращает фактическое число фиксируемых радиочастотных меток, соответствующих критериям. Если для аргумента Identify-Method (метод идентификации) установлено значение Inventory-No-More-Than (инвентаризация не более указанного числа радиочастотных меток), тогда значение в поле Number-Of-Tags-Found (число обнаруженных радиочастотных меток) может быть более низким.

7.5.11 Таблица идентификаторов упакованных объектов (PO-ID-Table)

Поле с именем PO-ID-Table (таблица идентификаторов упакованных объектов) — это поле команды, которая идентифицирует конкретную таблицу для процессора данных для использования в качестве источника подробных правил кодирования.

7.5.12 Контрольное слово протокола (Protocol-Control-Word)

Поле с именем Protocol-Control-Word (контрольное слово протокола) — это поле команды-ответа, которое возвращает это 16-битовое значение в ответе из памяти идентификатора UII радиочастотной метки с сегментированной памятью. Поле Protocol-Control-Word (контрольное слово протокола) содержит битовые данные, которые идентифицируют другие функции, поддерживаемые радиочастотной меткой.

7.5.13 Считывание данных (Read-Data)

Поле с именем Read-Data (считывание данных) возвращает недекодированную строку байтов из радиочастотных меток с сегментированной памятью.

7.6 Обеспечение безопасности данных (Data security)

Данные (объект) могут быть защищены с помощью алгоритмов шифрования. Применение алгоритмов шифрования должно происходить до передачи объекта в процессор данных (data processor). Процесс расшифрования должен применяться к объекту уже после его передачи в приложение. Таким образом, использование криптографических механизмов будет прозрачным по отношению к настоящему стандарту и ИСО/МЭК 15962.

Примечание — Дополнительные рекомендации приведены в приложении D.

8 Потоки данных и процессы для радиоинтерфейса

8.1 Общие положения

Различные процессы требуются, например, для форматирования радиочастотной метки, записи на нее данных, считывания с нее, изменения данных и т. д. Они определены в нижеприведенных подразделах. В них приведены все процессы для записи и добавления данных. Если процесс считывания является обратным для процесса записи, это описывается кратко, в противном случае предоставляется дополнительное описание.

8.2 Установление связи между приложением и радиочастотной меткой

8.2.1 Общие положения

Процессор данных (data processor) не связывается напрямую с радиочастотной меткой, а делает это через драйвер радиочастотных меток (Tag Driver). Процессор данных требует определенной системной информации на основе конфигурации радиочастотной метки (см. 8.2.3). Это означает, что параметры логической памяти надлежащим образом отображают память радиочастотной метки и позволяют осуществлять двустороннюю связь с драйвером радиочастотных меток. С этой целью для установления связи процессору данных должны предоставляться через драйвер радиочастотных меток сервисы радиоинтерфейса (см. 8.2.2).

Ряд этих параметров должен быть известен процессору данных или запрошен им. Фактически эта процедура используется для первоначального конфигурирования радиочастотной метки, для ее повторного конфигурирования, если требуется, и для установления связи, когда транзакция данных открыта.

8.2.2 Сервисы радиоинтерфейса

Настоящий стандарт является открытым с той точки зрения, что в серию стандартов ИСО/МЭК 18000* могут быть добавлены новые типы радиочастотной метки и при этом процессор данных (data processor) останется неизменным. Для достижения этой цели сделаны некоторые основные предположения о типах радиочастотной метки в серии стандартов ИСО/МЭК 18000*:

- память приложения (application memory) представляет собой целое число байтов.

Примечание — Термин «память приложения» (application memory) используется в этой подкатегории как общее имя для области памяти радиочастотной метки, доступной для пользовательских данных (в серии стандартов ИСО/МЭК 18000* это иногда называют пользовательской памятью), и любой отдельно адресуемой памяти (например, для идентификатора UII);

- память приложения должна быть организована в виде блоков. Блоки должны быть фиксированного размера и представлять собой один или несколько байтов;
- в дополнение к вышеуказанным требованиям, относящимся к памяти, существует надежный механизм для записи и считывания из памяти.

Радиочастотная метка должна иметь механизм для хранения системной информации (см. 8.2.3), включая возможность записи и считывания составных элементов (component elements).

Подробные технические сведения о сервисах радиоинтерфейса приведены в ИСО/МЭК 15962.

8.2.3 Системная информация

Системная информация должна состоять из следующих элементов, которые должны быть переданы через прикладной интерфейс и радиоинтерфейс и поэтому являются частью настоящего стандарта и ИСО/МЭК 15962:

- идентификатор Singulation-Id (см. 7.2.1);
- идентификатор AFI (см. 7.2.2);
- идентификатор DSFID (см. 7.2.3), который сам состоит:
  - из метода доступа (Access-Method) (см. 7.2.4);
  - формата данных (Data-Format) (см. 7.2.5).

8.3 Службы прикладной системы

В прикладной системе (application system) должны быть предусмотрены:

- идентификатор объекта (Object Identifier) (см. 7.3.3);
- объект (Object) (см. 7.3.5);

---

* *См.* [*10*] — [*18*].

- параметр уплотнения (Compact-Parameter) (см. 7.3.6);
- блокировка объекта (Object-Lock) (см. 7.3.7).

Они включены в определения конкретных команд и ответов приложений и никогда не выполняются без вспомогательных команд.

9 Коды команд, коды завершения и коды выполнения

9.1 Общие положения

Команды приложения используются для того, чтобы дать указание процессору данных (data processor) и устройству считывания выполнить определенные функции. Они также применяются для обработки данных приложения с целью достижения эффективного кодирования. Ответы от процессора данных включают запрошенные данные, а также информацию о предпринятых действиях и обнаруженных ошибках. Каждая пара команда — ответ имеет свои абстрактные синтаксисы, представленные в виде модулей. Каждый модуль формируется таким образом, чтобы команда могла быть вызвана независимо от любой другой команды.

Чтобы команды и ответы легко могли быть включены в синтаксис передачи, в настоящем стандарте конечной дуге модулей команд и ответов присвоены кодовые значения (см. 9.2). Ответам присвоены кодовые значения завершения (Completion-Codes) (см. 9.3) и кодовые значения выполнения (Execution-Codes) (см. 9.4). Заданы источники определений аргументов и полей, которые применяются к каждой команде, включая те, что применяются только к модулям команд и ответов (см. 7.4 и 7.5).

При обработке команды может быть обнаружена ошибка. Команда должна быть прервана, и при этом в радиочастотную метку, в которой обнаружена ошибка, или из нее не передаются никакие данные. Это возможно потому, что обработка выполняется в логической памяти. Несмотря на возможность присутствия и других ошибочных ситуаций, первая идентифицированная проблема является единственной, о которой и сообщается. Возвращается соответствующее кодовое значение завершения (Completion-Code) или кодовое значение выполнения (Execution-Code). В случае, когда команда обращается к нескольким радиочастотным меткам, должны быть обработаны все радиочастотные метки, обработанные до радиочастотной метки с обнаруженной ошибкой. Обнаружение ошибки прерывает всю последующую обработку.

Следующие подразделы определяют все команды и ответы приложений, которые поддерживаются настоящим стандартом. В дополнение к основному синтаксису эти подразделы будут также описывать функцию и назначение конкретных аргументов команд и ответов. Команды и ответы группируются логически вместе. В этом разделе показан текущий надлежащий абстрактный синтаксис.

В приложении Е приведены исходные 16 модулей с использованием абстрактного синтаксиса АСН.1 и терминология, используемая в ИСО/МЭК 15961:2004*. Некоторые из этих модулей имеют прямые эквиваленты в текущем формате, другие были объединены для создания новых модулей, и это приведено в соответствующих подразделах. В приложении F приведен пример исходного кодирования передачи команды и ответа.

9.2 Значения конечных дуг модулей команд и ответов

Каждый модуль команды и ответа должен быть распознан с помощью идентификатора объекта (Object Identifier). Общим корнем для команд является {iso(1) standard(0) rfid-data-protocol(15961) commandModules(126)}. Общим корнем для ответов является {iso(1) standard(0) rfid-data-protocol(15961) commandResponses(127)}. Конечная дуга каждой пары модулей команд и ответов должна иметь одинаковое значение, что фактически является относительным идентификатором объекта (Relative-OID). Завершающая дуга должна быть специфичной для пары команд/ответов и следующих указанных конечных дуг:

1 — сконфигурировать идентификатор AFI (Configure-AFI);
2 — сконфигурировать идентификатор DSFID (Configure-DSFID);
3 — инвентаризировать радиочастотные метки (Inventory-Tags);
4 — зарезервировано (использовалось отмененным стандартом ИСО/МЭК 15961:2004*);
5 — удалить объект (Delete-Object);
6 — изменить объект (Modify-Object);
7 — зарезервировано (использовалось отмененным стандартом ИСО/МЭК 15961:2004*);
8 — считать идентификаторы объектов (ReadObject-Identifiers);

* *См.* [*8*].

9 — зарезервировано (использовалось отмененным стандартом ИСО/МЭК 15961:2004)*;
10 — считать карту логической памяти (Read-Logical-Memory-Map);
11 — зарезервировано (использовалось отмененным стандартом ИСО/МЭК 15961:2004*);
12 — очистить память (Erase-Memory);
13 — получить системную информацию, связанную с приложением (Get-App-Based-System-Info);
14 — зарезервировано (использовалось отмененным стандартом ИСО/МЭК 15961:2004*);
15 — зарезервировано (использовалось отмененным стандартом ИСО/МЭК 15961:2004*);
16 — зарезервировано (использовалось отмененным стандартом ИСО/МЭК 15961:2004*);
17 — записать объекты (Write-Objects);
18 — считать объекты (Read-Objects);
19 — записать объекты в сегментированную память радиочастотной метки (Write-Objects-Segmented-Memory-Tag);
20 — записать EPC-UII (Write EPC-UII);
21 — инвентаризировать память ISO-UII (Inventory-ISO-UIImemory);
22 — инвентаризировать память EPC-UII (Inventory-EPC-UIImemory);
23 — записать пароль в сегментированную память радиочастотной метки (Write-Password-Segmented-Memory-Tag);
24 — считать слова из сегментированной памяти радиочастотной метки (Read-Words-Segmented-Memory-Tag);
25 — уничтожить сегментированную память радиочастотной метки (Kill-Segmented-Memory-Tag);
26 — удалить упакованный объект (Delete-Packed-Object);
27 — изменить упакованный объект (Modify-Packed-Object);
28 — записать сегменты радиочастотной метки типа 6TypeD (Write-Segments-6TypeD-Tag);
29 — считать сегменты радиочастотной метки типа 6TypeD (Read-Segments-6TypeD-Tag);
30 — записать мономорфный идентификатор UII (Write-Monomorphic-UII);
31 — сконфигурировать расширенный идентификатор DSFID (Configure-Extended-DSFID);
32 — сконфигурировать заголовок блочных записей (Configure-Multiple-Records-Header);
33 — считать блочные записи (Read-Multiple-Records);
34 — удалить блочную запись (Delete-Multiple-Record).

Дополнительные модули команд и ответов, а также значения их конечных дуг будут добавлены, если это требуется, в числовой последовательности в настоящий стандарт.

Заполненный модуль определяет функцию, которую должно выполнять устройство опроса. Каждая команда указывает, при необходимости, процессы, которые должны выполняться процессором данных (data processor), драйвером радиочастотных меток и устройством опроса в сообщениях по радиоинтерфейсу. Каждый ответ указывает, в зависимости от ситуации, процессы, которые должны быть выполнены драйвером радиочастотных меток, процессором данных и системой коммуникаций через интерфейс приложения.

9.3 Код завершения (Completion-Code)

Completion-Code (код завершения) является частью ответа на каждую команду. Completion-Code (код завершения) имеет значение INTEGER (целочисленное значение) и сообщает о том, как определенная команда была обработана и выполнена, успешно или нет. Он возвращается в каждом ответе. Если его значение равно 0 (00_{16}), то команда была успешно выполнена. Если его значение равно 255 (FF_{16}), то команда не может быть выполнена системой по причине, указанной в коде выполнения (см. 9.4). Если его значение отличается от 0 и от 255, то это означает, что команда не была выполнена, как сообщается приложением, по указанной причине.

Примечание — Completion-Code (код завершения) содержит информацию о том, что команда может быть вызвана для конкретной радиочастотной метки в цепи связи, тогда как Execution-Code (код выполнения) указывает на системную ошибку или надлежащее выполнение.

Должны применяться кодовые значения завершения, указанные в ИСО/МЭК 15962.

9.4 Код выполнения (Execution-Code)

Execution-Code (код выполнения) является частью ответа на каждую команду. Execution-Code (код выполнения) имеет значение INTEGER (целочисленное значение) и сообщает о том, как команда была

* *См.* [*8*].

обработана и выполнена системой, успешно или нет. Он возвращается в каждом ответе. Если его значение равно 0 (00_{16}), то команда была успешно обработана, то есть протокол был выполнен. Другие значения указывают на системную ошибку.

Должны применяться кодовые значения выполнения, указанные в ИСО/МЭК 15962.

10 Команды и ответы на команды

10.1 Общие положения

Каждая пара команд и ответов определяется в 10.2—10.22. Те аргументы, которые требуют отдельной спецификации, определены в 10.28.2.

10.2 Сконфигурировать идентификатор AFI (Configure-AFI)

10.2.1 Команда Configure-AFI (сконфигурировать идентификатор AFI)

Идентификатор семейства приложений (AFI) является однобайтовым кодом и используется как часть процесса выбора в приложении. Подробная информация о кодовых значениях AFI, присвоенных конкретным приложениям, доступна в реестре конструкций данных, предоставленном органом регистрации по ИСО/МЭК 15961-2:2019*. Если радиоинтерфейс поддерживает функциональность идентификатора AFI, то для последующей обработки будут возвращены только радиочастотные метки с конкретным идентификатором AFI.

Эта команда применима для тех протоколов радиоинтерфейса, где истинна одна из следующих характеристик:

A. команда приложения напрямую связана с эквивалентной командой радиоинтерфейса;

B. команда приложения требует, чтобы идентификатор AFI записывался в определенную ячейку памяти и номинально был отделен от данных, связанных с предметами, с использованием стандартной команды записи радиоинтерфейса.

Команда Configure-AFI (сконфигурировать идентификатор AFI) имеет следующие аргументы:

- AFI (идентификатор AFI) (см. 7.2.2);
- AFI-Lock (блокировка идентификатора AFI) (см. 7.4.3);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Аргумент AFI-Lock (блокировка идентификатора AFI) применим к характеристике A, указанной выше, без ограничений, но может применяться к вышеуказанной характеристике B, только если протокол радиоинтерфейса поддерживает блокировку положения единственного байта, определенного для идентификатора AFI.

Команда Configure-AFI (сконфигурировать идентификатор AFI)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 1
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
AFI (идентификатор AFI): BYTE (БАЙТ)
Возможные значения:

Значение	Определение
00_{16} — $0F_{16}$	Как определено в ИСО/МЭК 15961-3
90_{16} — CE_{16}	Как опубликовано органом регистрации по ИСО/МЭК 15961-2:2019*
CF_{16}	Зарезервировано как код расширения (extension code)

AFI-Lock (блокировка идентификатора AFI): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)
Если установлено значение TRUE (ИСТИНА), устройство опроса должно заблокировать идентификатор AFI

10.2.2 Ответ на команду Configure-AFI (сконфигурировать идентификатор AFI)

Ответ на команду Configure-AFI (сконфигурировать идентификатор AFI) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.2);
- Execution-Code (код выполнения) (см. 9.3).

* *См.* [*9*].

Ответ на команду Configure-AFI (сконфигурировать идентификатор AFI)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 1
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
1	AFI-Not-Configured (идентификатор AFI несконфигурирован)
2	AFI-Not-Configured-Locked (блокировка несконфигурированного идентификатора AFI)
3	AFI-Configured-Lock-Failed (блокировка сконфигурированного идентификатора AFI не выполнена)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.3 Сконфигурировать идентификатор DSFID (Configure-DSFID)

10.3.1 Общие положения

Команда Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID) (см. 10.25) требуется при кодировании индикаторов других функций в радиочастотной метке, таких как использование циклического избыточного контрольного кода (CRC) данных. Эта команда также может лучше кодировать метод доступа (Access-Method) со значением больше 3 или формат данных (Data-Format) со значением, превышающим 31.

10.3.2 Команда Configure-DSFID (сконфигурировать идентификатор DSFID)

Идентификатор DSFID — это однобайтовый код, который используется для упрощения кодирования идентификатора объекта и определения конкретных правил кодирования для ИСО/МЭК 15962. Эти правила кодирования применяются к методу доступа (Access-Method). Подробная информация о форматах данных (Data-Format) (части идентификатора DSFID), присвоенных приложениям, приведена в Реестре конструкций данных (Register of Data Construct), предоставляемом органом регистрации (Registration Authority) по ИСО/МЭК 15961-2:2019*.

Команда применима для тех протоколов радиоинтерфейса, где подтверждена одна из следующих характеристик:

a) команда приложения напрямую связана с эквивалентной командой радиоинтерфейса;

b) команда приложения требует, чтобы идентификатор DSFID записывался в определенное место памяти, номинально отделенное от данных, связанных с предметами, но используя стандартную команду записи радиоинтерфейса.

Команда Configure-DSFID «(сконфигурировать идентификатор DSFID) имеет следующие аргументы:

- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2);
- DSFID-Lock (блокировка идентификатора DSFID) (см. 7.4.15);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Аргумент DSFID-Lock (блокировка идентификатора DSFID) применим к вышеуказанной характеристике a) без ограничений, но может применяться к характеристике b), только если протокол радиоинтерфейса поддерживает блокировку позиции единственного байта, определенного идентификатором DSFID.

Команда Configure-DSFID (сконфигурировать идентификатор DSFID)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 2
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
DSFID-Constructs-list (перечень значений конструкций идентификатора DSFID): List of <DSFID-Constructs> (перечень конструкций идентификатора DSFID)
DSFID-Lock (блокировка идентификатора DSFID): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)
Если установлено значение TRUE (ИСТИНА), устройство опроса должно заблокировать идентификатор DSFID

* *См.* [9].

10.3.3 Ответ на команду Configure-DSFID (сконфигурировать идентификатор DSFID)

Ответ на команду Configure-DSFID response (сконфигурировать идентификатор DSFID) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Configure-DSFID (сконфигурировать идентификатор DSFID)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 1
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
4	DSFID-Not-Configured (идентификатор DSFID не сконфигурирован)
5	DSFID-Not-Configured-Locked (блокировка несконфигурированного идентификатора DSFID)
6	DSFID-Configured-Lock-Failed (блокировка сконфигурированного идентификатора DSFID не выполнена)
8	Singulation-Id-Not-Found (индивидуализированный идентификатор не обнаружен)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.4 Инвентаризировать радиочастотные метки (Inventory-Tags)

10.4.1 Команда Inventory-Tags command (инвентаризировать радиочастотные метки)

Команда Inventory-Tags (инвентаризировать радиочастотные метки) требует указания значения идентификатора AFI для выбора радиочастотных меток, принадлежащих к определенному классу, обычно содержащему радиочастотные метки, принадлежащие определенному домену. Команда «инвентаризировать радиочастотные метки» (Inventory-Tags command) предназначена для считывания набора идентификаторов Singulation-Id из радиочастотных меток, которые имеют определенный идентификатор AFI. Это применимо только там, где команда радиоинтерфейса поддерживает процесс инвентаризации с использованием идентификатора AFI в качестве именованного аргумента и, как правило, где уникальный идентификатор микросхемы используется при проведении арбитража.

Дополнительный критерий выбора (метод идентификации) определяет, сколько радиочастотных меток, соответствующих критерию выбора определенного идентификатора AFI, необходимо идентифицировать до того, как будет предоставлен ответ на команду. Механизм, который может быть использован для обнаружения любой радиочастотной метки, входящей в рабочую область, заключается в том, чтобы установить значение поля Inventory-At-Least (инвентаризация не менее, чем) в 1. Конкретные условия могут быть подтверждены только за счет частичной инвентаризации, т. е. с использованием либо значения поля Inventory-At-Least (инвентаризация не менее, чем), либо Inventory-No-More-Than (инвентаризация не более, чем). Согласование известного числа предыдущих транзакций (например, для идентификации того, что все предметы, предназначенные для размещения в контейнере, действительно находятся там) может быть достигнуто с использованием значения поля Inventory-Exactly (инвентаризация в точном соответствии).

Команда Inventory-Tags (инвентаризировать радиочастотные метки) имеет следующие аргументы:

- AFI (идентификатор AFI) (см. 7.2.2);
- Identify-Method (метод идентификации) (см. 7.4.23);
- Number-Of-Tags (число радиочастотных меток) (см. 7.4.43).

Команда Inventory-Tags (инвентаризировать радиочастотные метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 3
AFI (идентификатор AFI): BYTE (БАЙТ)
Identify-Method (метод идентификации): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (0..15)

Возможные значения:

Значение	Определение
0	Inventory-All-Tags (инвентаризация всех радиочастотных меток)
1	Inventory-At-Least (инвентаризация не менее, чем)
2	Inventory-No-More-Than (инвентаризация не более, чем)
3	Inventory-Exactly (инвентаризация в точном соответствии)
4—15	Резерв

Number-Of-Tags (число радиочастотных меток): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (0..65535)

Если для аргумента Identify-Method (метод идентификации) установлено значение Inventory-All-Tags (инвентаризация всех радиочастотных меток), устройство опроса должно выполнить полную инвентаризацию всех радиочастотных меток, присутствующих в его зоне действия. Значение аргумента Number-Of-Tags (число радиочастотных меток) не существенно и должно быть установлено приложением в ноль.

Если для аргумента Identify-Method (метод идентификации) установлено значение Inventory-At-Least (инвентаризация не менее, чем), устройство опроса должно выполнить инвентаризацию радиочастотных меток, присутствующих в его зоне действия, и (возможно) продолжить ожидание до тех пор, пока оно не определит число радиочастотных меток, равное значению аргумента Number-Of-Tags (число радиочастотных меток). Если для аргумента Number-Of-Tags (число радиочастотных меток) установлено значение 1, устройство опроса будет ждать до тех пор, пока не обнаружится первая радиочастотная метка. Это механизм ожидания входа радиочастотной метки в зону действия устройства опроса. Если для аргумента установлено значение более 1, устройство опроса будет находиться в ожидании, пока не обнаружится установленное число радиочастотных меток.

Если для аргумента Identify-Method (метод идентификации) установлено значение Inventory-No-More-Than (инвентаризация не более, чем), устройство опроса должно инициировать инвентаризацию радиочастотных меток, присутствующих в его зоне действия, и вернуть ответ с числом радиочастотных меток, меньшим или равным значению аргумента Number-Of-Tags (число радиочастотных меток). Устройство опроса может прервать процесс инвентаризации, когда достигнуто данное значение Number-Of-Tags (число радиочастотных меток), или продолжить процесс инвентаризации до тех пор, пока не будут считаны все радиочастотные метки.

Примечание 1 — Могут быть ограничения со стороны радиоинтерфейса и антиколлизионного механизма.

Если для аргумента Identify-Method (метод идентификации) установлено значение Inventory-Exactly (инвентаризация в точном соответствии), устройство опроса должно инициировать инвентаризацию радиочастотных меток, присутствующих в его зоне действия, и вернуть ответ с числом радиочастотных меток, равным значению аргумента Number-Of-Tags (число радиочастотных меток). Этот параметр команды допускается использовать для подтверждения фактического числа предметов, снабженных радиочастотными метками, находящихся в контейнере. Устройство опроса будет ожидать, пока не обнаружится установленное число радиочастотных меток. Устройство опроса может прервать процесс инвентаризации, когда достигнуто значение Number-Of-Tags (число радиочастотных меток), или продолжать процесс инвентаризации до тех пор, пока не будут прочитаны все радиочастотные метки.

Примечание 2 — Могут быть ограничения со стороны радиоинтерфейса и антиколлизионного механизма.

Выполнение этой команды с аргументами Inventory-At-Least (инвентаризация не менее, чем) и Inventory-Exactl (инвентаризация в точном соответствии) может заставить устройство опроса ожидать, пока в зону его действия не войдет достаточное число радиочастотных меток. Команда-ответ также не может быть инициирована до истечения этой задержки. Приложение несет ответственность за реализацию этой потенциальной возможности.

10.4.2 Ответ на команду Inventory-Tags response (инвентаризировать радиочастотные метки)

Ответ на команду Inventory-Tags response (инвентаризировать радиочастотные метки) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- Identities (ключевые идентификаторы) (см. 7.5.3);
- Number-Of-Tags-Found (число обнаруженных радиочастотных меток) (см. 7.5.10).

Ответ на команду Inventory-Tags (инвентаризировать радиочастотные метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) =1 0 15961 127 3
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
23	Failed-To-Read-Minimum-Number-Of-Tags (не удалось считать минимальное число радиочастотных меток)
24	Failed-To-Read-Exact-Number-Of-Tags (не удалось считать точное число радиочастотных меток)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

Number-Of-Tags-Found (число обнаруженных радиочастотных меток): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (1…65535)
Identities (ключевые идентификаторы): List of <Singulation-Id> (список идентификаторов <Singulation-Id>)

10.5 Удалить объект (Delete-Object)

10.5.1 Команда Delete-Object (удалить объект)

Команда Delete-Object (удалить объект) дает указание устройству опроса на удаление определенного идентификатора объекта (Object-Identifier), его объекта (Object) и связанных с ним параметров. Чтобы гарантировать надежность процесса удаления, для каждой команды необходимо запрограммировать только одну радиочастотную метку и только один идентификатор объекта. Для функции удаления требуется удаление идентификатора объекта, связанного с ним объекта, прекурсора и других компонентов набора данных с карты логической памяти, а затем перезапись всех наборов данных в ячейках радиочастотной метки в ячейки памяти с более высокими адресами.

Для методов доступа No-Directory (без каталога) и Directory (каталог) процессор данных (data processor) может заменить удаленные байты нулевым набором данных, если другое кодирование следует за удаленным идентификатором объекта. Эта процедура автоматически вызывается процессором данных (data processor).

Для метода доступа Packed-Objects (упакованные объекты) устройство опроса должно удалить объект из первого упакованного объекта, в котором существует идентификатор объекта. Если упакованный объект доступен для редактирования, процедуры редактирования упакованного объекта должны использоваться так, чтобы не требовалось перезаписывать весь объем памяти. В противном случае устройство опроса должно перезаписать содержимое пользовательской памяти, соответствующим образом пересчитанное после удаления идентификатора объекта и его объекта, а также связанных с ним параметров. Если какая-либо область памяти, которую требуется перезаписать во время выполнения этой команды, оказывается заблокированной, в ответе будет возвращен соответствующий Completion-Code (код завершения).

Если методом доступа является Tag-data-Profile (профиль данных радиочастотных меток), удаление невозможно из-за фиксированной структуры кодирования и того факта, что системные нулевые знаки отсутствуют, и должен быть возвращен соответствующий Completion-Code (код завершения).

Для блочных записей (Multiple-Records) идентификатор объекта (Object-Identifier) должен идентифицировать элемент данных в отдельной записи. Для используемого метода доступа должны применяться соответствующие правила. Для удаления целиком всей записи см. 10.28.

Команда Delete-Object (удалить объект) дает указание устройству опроса удалить набор данных, определенный его идентификатором объекта (Object-Identifier), из карты логической памяти радиочастотной метки. Эта процедура может не выполниться, если набор данных заблокирован. Если это действительно так, в ответе вернется соответствующий Completion-Code (код завершения). Если значение флага Check-Duplicate (проверка дубликата) установлено в TRUE (ИСТИНА), устройство опроса перед удалением запрошенного объекта должно проверить, что существует только один экземпляр

запрашиваемого идентификатора объекта. Если устройство опроса обнаруживает, что радиочастотная метка кодирует более одного экземпляра ссылочного идентификатора объекта, он не должен выполнять функцию Delete-Object (удалить объект) и должен вернуть соответствующий Completion-Code (код завершения).

Если значение флага Check-Duplicate (проверка дубликата) установлено в значение FALSE (ЛОЖЬ), устройство опроса должно удалить первое вхождение набора данных, указанного его идентификатором объекта (Object-Identifier).

Этот аргумент фактически не обеспечивает защиту от дублирующих идентификаторов объекта. Его следует использовать только тогда, когда существует большая вероятность отсутствия дубликатов.

Команда Delete-Object (удалить объект) имеет следующие аргументы:

- Check-Duplicate (проверка дубликата) (см. 7.4.10);
- Object-Identifier (идентификатор объекта) (см. 7.3.3);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Delete Object (удалить объект)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 5
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)
Check-Duplicate (проверка дубликата): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), устройство опроса должно проверить, что есть только одно появление идентификатора объекта

10.5.2 Ответ на команду Delete-Object (удалить объект)

Ответ на команду Delete-Object (удалить объект) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Delete-Object (удалить объект)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 5
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
10	Duplicate-Object (дубликат объекта)
12	Object-Not-Deleted (объект не удален)
13	Object-Identifier-Not-Found (идентификатор объекта не обнаружен)
14	Object-Locked-Could-Not-Delete (блокированный объект не может быть удален)
37	Data-CRC-Not-Applied (код CRC к данным не применяется)
38	Length-Not-Encoded-In-DSFID (длина не закодирована в идентификаторе DSFID)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.6 Изменить объект (Modify-Object)

10.6.1 Команда Modify-Object (изменить объект)

Команда Modify-Object (изменить объект) предназначена для изменения значения объекта данных, связанного с идентификатором объекта, уже закодированного в памяти радиочастотной метки. Чтобы идентификатор объекта не дублировался, память должна быть считана целиком. Если этот факт обнаружился, команда прерывается. Вызов данной команды зависит от метода доступа (Access-Method), объявленного для радиочастотной метки. Кроме того, процедура отличается, если результирующая длина кодирования измененного объекта отличается от исходной длины. Каждый из этих случаев обсуждается в рамках соответствующего метода доступа.

Эта команда не должна использоваться для изменения мономорфного идентификатора UII (Monomorphic-UII). Если идентификатор AFI в радиочастотной метке объявляет, что он зарегистрирован для идентификатора Monomorphic-UII, возвращается соответствующая ошибка, и процесс кодирования прерывается. Надлежащая команда для использования определена в 10.24.

Команда Modify-Object (изменить объект) дает указание устройству опроса выполнить три связанных процесса:

a) считать полную карту логической памяти из радиочастотной метки;

b) идентифицировать закодированный пакет [например, Data-Set (набор данных), или Packed-Object (упакованный объект), или Tag-Data-Profile (профиль данных радиочастотных меток)], указанный идентификатором объекта (Object-Identifier). Если обнаружены дублирующие экземпляры, процесс прерывается;

c) выполнить перезапись с измененным кодированным пакетом:

- включая реструктуризацию прекурсора для Data-Set (набора данных),
- включая любые правила структурирования Packed-Object (упакованный объект),
- включая любые заполняющие байты для более коротких данных в Tag-Data-Profile (профиль данных радиочастотных меток).

Если объект уже заблокирован, его невозможно изменить и получить соответствующий Completion-Code (код завершения).

Если метод доступа (Access-Method) применяется к упакованным объектам (Packed-Objects) и упакованный объект недоступен для редактирования, объект не может быть изменен и должен быть возвращен соответствующий Completion-Code (код завершения).

Если метод доступа представляет собой Tag-Data-Profile (профиль данных радиочастотных меток), то могут быть рассмотрены три условия:

A. если новые уплотненные данные имеют одинаковую длину, то эти данные просто перезаписываются;

B. если они короче, они записываются в радиочастотную метку с необходимым числом заполняющих байтов;

C. если они длиннее, то присутствует ошибка; данные не могут быть изменены, и должен быть возвращен соответствующий Completion-Code (код завершения).

Если объект данных является частью блочной записи [идентифицированной с базовым корневым идентификатором объекта (root-OID) 1.0.15961.401] или иерархической блочной записью [идентифицированной с базовым корневым идентификатором объекта (root-OID) 1.0.15961.402], то процесс изменения объекта данных должен соответствовать объявленному методу доступа. Объект данных не может быть изменен в списке элементов данных [идентифицирован с базовым корневым идентификатором объекта (root-OID) 1.0.15961.403].

Команда Modify-Object (изменить объект) имеет следующие аргументы:

- Compact-Parameter (параметр уплотнения) (см. 7.3.6);
- Object (объект) (см. 7.3.5);
- Object-Identifier (идентификатор объекта) (см. 7.3.3);
- Object-Lock (блокировка объекта) (см. 7.3.7);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Modify-Object (изменить объект)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 6
Singulation-Id (идентификатор Singulation-Id): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (0..255)
Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)
Object (объект): BYTE STRING (СТРОКА БАЙТОВ)
Compact-Parameter (параметр уплотнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (0..15)

Возможные значения:

Значение	Определение (см. 7.3.6 для получения подробной информации)
0	Application-Defined (определено приложением)
1	Compact (уплотнение)
2	UTF8-Data (данные в формате UTF-8)
3	Packed-Objects (упакованные объекты)
4	Tag-Data-Profile (профиль данных радиочастотных меток)

Object-Lock (блокировка объекта): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если значение TRUE (ИСТИНА), то устройство опроса должно заблокировать соответствующий Data-Set (набор данных).

10.6.2 Ответ на команду Modify-Object (изменить объект)

Ответ на команду Modify-Object (изменить объект) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Modify-Object (изменить объект)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 6
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (Нет ошибки)
7	Object-Locked-Could-Not-Modify (блокированный объект не может быть изменен)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
10	Duplicate-Object (дубликат объекта)
13	Object-Identifier-Not-Found (идентификатор объекта не обнаружен)
21	Object-Not-Modified (объект не изменен)
22	Object-Modified-But-Not-Locked (объект изменен, но не заблокирован)
33	Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)
36	Command-Cannot-Process-Monomorphic-UII (команда не может обрабатывать моно-морфный идентификатор UII)
37	Data-CRC-Not-Applied (код CRC к данным не применяется)
38	Length-Not-Encoded-In-DSFID (длина не закодирована в идентификаторе DSFID)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.7 Считать идентификаторы объектов (Read-Object-Identifiers)

10.7.1 Команда Read-Object-Identifiers (считать идентификаторы объектов)

Эта команда не должна использоваться с блочными записями. Вместо этого должна использоваться команда Read-Multiple-Records (считать блочные записи) (см. 10.27).

Команда Read-Object-Identifiers (считать идентификаторы объектов) дает указание устройству опроса на считывание всех идентификаторов объектов (Object-Identifiers) из радиочастотной метки. Этот модуль допускается использовать перед более избирательной командой для считывания определенного объекта или для идентификации дублирующихся идентификаторов объектов. Если на карте логической памяти радиочастотной метки нет идентификаторов объектов, надлежащим ответом является возвращение пустого списка идентификаторов объектов. Для каждой команды должна быть запрограммирована только одна радиочастотная метка, чтобы гарантировать надежность процесса считывания.

Команда Read-Object-Identifiers (считать идентификаторы объектов) имеет следующий аргумент:

- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1)

Команда Read-Object-Identifiers (считать идентификаторы объектов)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 8
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0…255)

10.7.2 Ответ на команду Read-Object-Identifiers (считать идентификаторы объектов)

Ответ на команду Read-Object-Identifiers (считать идентификаторы объектов) имеет следующие аргументы:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- Read-OIDs-Response-List (список ответов на считывание идентификаторов объектов) (см. 11.15).

Ответ на команду Read-Object-Identifiers (считать идентификаторы объектов) Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 127 8

Read-OIDs-Response-List (список ответов на считывание идентификаторов объектов): List of <Read-OIDs-Response> (список <ответы на команду «считать идентификаторы объектов»>)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.8 Считать карту логической памяти (Read-Logical-Memory-Map)

10.8.1 Команда Read-Logical-Memory-Map (считать карту логической памяти)

Основная функция этой команды предназначена для целей диагностики, но ее также допускается использовать для других функций, где требуется считывание полного содержимого карты логической памяти. Для каждой команды необходимо запрограммировать только одну радиочастотную метку, чтобы гарантировать надежность процесса считывания.

Команда «считать карту логической памяти» (Read-Logical-Memory-Map) дает указание устройству опроса считывать карты логической памяти радиочастотной метки и ответить на это без какого-либо декодирования и интерпретации (т. е. путем возврата значений закодированного байта). Обработка данных через процессор данных (data processor) не является частью этой команды считывания, поэтому отдельные идентификаторы объектов (Object-Identifiers), объекты (Objects), параметр уплотнения (Compact-Parameter) и статус блокировки (Lock-Status) не могут быть идентифицированы напрямую.

Команда применяется в равной степени ко всем методам доступа, но, если структура каталога была определена методом доступа, это должно быть включено в ответ, но не должно отличаться от других байтов на карте логической памяти.

Команда Read-Logical-Memory-Map (считать карту логической памяти) имеет следующий аргумент:

- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Read-Logical-Memory-Map (считать карту логической памяти)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 10
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0…255)

10.8.2 Ответ на команду Read-Logical-Memory-Map (считать карту логической памяти)

Ответ на команду Read-Logical-Memory-Map (считать карту логической памяти) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- Logical-Memory-Map (логическая карта памяти) (см. 7.5.7).

Ответ на команду Read-Logical-Memory-Map (считать карту логической памяти)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 10
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
19	Read-Incomplete (неполное считывание)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

Logical-Memory-Map (карта логической памяти): BYTE STRING (СТРОКА БАЙТОВ)

10.9 Очистить память (Erase-Memory)

10.9.1 Команда Erase-Memory (очистить память)

Команда Erase-Memory (очистить память) дает указание устройству опроса произвести повторное полное обнуление карты логической памяти определенной радиочастотной метки. Очистка включает в себя Directory (каталог), если он определен как метод доступа (Access-Method). Если ни один из блоков не заблокирован, это должно привести к удалению всех наборов данных (Data-Sets) или упакованных объектов (Packed-Objects). Если какой-либо блок заблокирован, то возвращается Completion-Code (код завершения): со значением Blocks-Locked (блоки заблокированы). Для каждой команды необходимо запрограммировать только одну радиочастотную метку, чтобы гарантировать надежность процесса очистки.

Для того, чтобы определить, по-прежнему ли используется радиочастотная метка, может быть вызвана последующая обработка с помощью приложения, возможно, путем считывания всех заблокированных данных. Например, заблокированные блоки могут содержать данные, постоянно назначенные данному предмету, и разблокированные блоки, содержащие временные данные.

Команда Erase-Memory (очистить память) имеет следующий аргумент:

- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Erase-Memory (очистить память)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 12
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)

10.9.2 Ответ на команду Erase-Memory (очистить память)

Ответ на команду Erase-Memory (очистить память) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Erase-Memory (очистить память)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 12
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
17	Blocks-Locked (блоки заблокированы)
18	Erase-Incomplete (неполная очистка)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.10 Получить системную информацию, связанную с приложением (Get-App-Based-System-Info)

10.10.1 Команда Get-App-Based-System-Info (получить системную информацию, связанную с приложением)

Команда Get-App-Based-System-Info (получить системную информацию, связанную с приложением) дает указание устройству опроса на считывание системной информации и возвращение тех аргументов, которые имеют отношение к приложению, а именно идентификатор семейства приложений (AFI) и идентификатор формата хранения данных (DSFID). Эта команда полезна для типов радиочастотных меток, которые не возвращают эти коды как часть ответа на другие команды.

Команда Get-App-Based-System-Info (получить системную информацию, связанную с приложением) имеет следующий аргумент:

- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Get-App-Based-System-Info (получить системную информацию, связанную с приложением)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 13
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)

10.10.2 Ответ на команду Get-App-Based-System-Info (получить системную информацию, связанную с приложением)

Ответ на команду Get-App-Based-System-Info (получить системную информацию, связанную с приложением) имеет следующие аргументы:

- AFI (идентификатор AFI) (см. 7.2.2);
- Completion-Code (код завершения) (см. 9.3);
- DSFID (идентификатор DSFID) (см. 7.2.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Get-App-Based-System-Info (получить системную информацию, связанную с приложением)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 13
AFI (идентификатор AFI): BYTE (БАЙТЫ)
DSFID (идентификатор DSFID): BYTE STRING (СТРОКА БАЙТОВ)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
20	System-Info-Not-Read (системная информация не считывается)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.11 Записать объекты (Write-Objects)

10.11.1 Команда Write-Objects (записать объекты)

Команда Write-Objects (записать объекты) используется для записи одного или нескольких идентификаторов объектов и связанных с ними объектов в радиочастотную метку. Эта команда может быть реализована для записи исходных данных в радиочастотную метку или для добавления данных в метку. Команда поддерживается составным аргументом «список добавленных объектов» (Add-Objects-List).

Эта команда не должна использоваться для записи мономорфного идентификатора UII. Если идентификатор AFI в радиочастотной метке объявляет, что он зарегистрирован для мономорфного идентификатора UII, возвращается соответствующая ошибка и процесс кодирования прерывается. Надлежащая команда для использования определена в 10.24.

Список конструкций идентификатора DSFID (DSFID-Constructs-list) используется для указания метода доступа (Access-Method) и формата данных (Data-Format). Список конструкций расширенного идентификатора DSFID (Ext-DSFID-Constructs-list) используется для установки индикаторов для расширенных идентификаторов DSFID (Extended-DSFID) и указания процессору данных (data processor) на выполнение определенных процедур, например для применения к данным CRC (избыточного циклического кода).

Список конструкций идентификатора DSFID (DSFID-Constructs-list) и список конструкций расширенного индентификатора DSFID (Ext-DSFID-Constructs-list) предоставляются для использования одним из следующих способов:

- если данные записываются в пустую радиочастотную метку, они предоставляются как часть этой команды, чтобы минимизировать коммуникации;
- если данные добавляются к радиочастотной метке, то идентификатор DSFID в команде должен соответствовать идентификатору DSFID, уже закодированному в радиочастотной метке, иначе возникает ошибка, и процесс кодирования может прекратиться до того, как будут обработаны значительные объемы данных;
- кроме того, должны быть обработаны все требования, объявленные аргументами в списке конструкций расширенного идентификатора DSFID (Ext-DSFID-Constructs-list).

Аргумент DSFID-Lock (блокировка идентификатора DSFID), если он установлен, применяется ко всей строке байтов идентификатора DSFID и расширенного идентификатора DSFID (Extended-DSFID).

Если методом доступа (Access-Method) является Packed-Objects (упакованные объекты), все объекты, указанные в качестве аргументов, должны быть включены в один и тот же новый упакованный объект, добавленный после всех существующих в памяти упакованных объектов. Ряд аргументов применим только к упакованным объектам, и они определены в аргументе Packed-Object-Constructs (конструкции упакованного объекта).

Если методом доступа (Access-Method) является Tag-Data-Profile (профиль данных радиочастотных меток), все объекты, указанные в качестве аргументов, должны быть включены в один и тот же Tag-Data-Profile (профиль данных радиочастотных меток). Добавление объекта, не заданного аргументом Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток), следует рассматривать как ошибку, и при этом никакого кодирования не происходит.

Блочная запись (Multiple-Record) объявляется с помощью идентификаторов объектов (Object-Identifiers) в списке добавленных объектов (Add-Objects-List), которые присутствуют в одной из трех обязательных форм:

- 1.0.15961.401.{data format = dictionary}.{sector identifier}.{record type}.{instance-of}.{Relative-OID of data element} (1.0.15961.401.{формат данных = словарь}.{идентификатор сектора}.{тип записи}.{экземпляр}.{относительный идентификатор элемента данных});
- 1.0.15961.402.{data format = dictionary}.{sector identifier}.{record type}.{hierarchical id}.{ Relative-OID of data element} (1.0.15961.402.{формат данных = словарь}.{идентификатор сектора}.{тип записи}.{иерархический идентификатор}.{относительный идентификатор элемента данных};
- 1.0.15961.403.{data format = dictionary}.{sector identifier}.{record type}.{hierarchical id}.{ Relative-OID of data element}.{list element number} (1.0.15961.403.{формат данных = словарь}.{идентификатор сектора}.{тип записи}.{иерархический идентификатор}.{относительный идентификатор элемента данных}.{номер элемента списка}.

Аргумент Add-Objects-List (список добавленных объектов) применяется к одной записи в структуре блочных записей. Поэтому в структурах идентификаторов объектов, перечисленных выше, только значения конечной дуги допускают различие в одной команде.

Индивидуальные записи должны быть записаны только после создания запроса на сопровождение заголовка блочной записи.

Аргумент DSFID-Constructs-List (список конструкций идентификатора DSFID) должен использоваться для объявления того, что отдельная блочная запись соответствует правилам кодирования одного из следующих методов доступа (Access-Methods):

0 No-Directory (без каталога);

1 Not assigned (не задан);

2 Packed-Objects (упакованные объекты);

3 Tag-Data-Profile (профиль данных радиочастотных меток).

Для блочных записей команде требуются аргументы Multiple-Records-Constructs-List (список конструкций блочных записей) (см. 11.8). Она включает инструкции по резервированию памяти для увеличения размера записи, чтобы показать, требуется ли позиция в каталоге, а в некоторых случаях — чтобы определить, существует ли между этой записью и другими отношение «родитель—потомок». Существует также аргумент для объявления, определена ли эта запись как список элементов данных, и в этом случае аргумент Add-Objects-List (список добавленных объектов) содержит несколько экземпляров одного и того же относительного идентификатора объекта (Relative-OID). Команда также может использоваться для добавления элементов данных в существующую блочную запись, объявляя это в соответствующем аргументе Multiple-Records-Constructs-List (список конструкций блочных записей).

Параметр Ext-DSFID-Constructs-List (список конструкций расширенного идентификатора DSFID) также необходим для определения всех закодированных требований для блочной записи.

Заголовок блочной записи (MR-header) определяет правила для формата данных (Data-Format), метода доступа (Access-Method) и идентификатора сектора, которые определяют последующее кодирование отдельных записей. В зависимости от индивидуальных настроек правила требуют, чтобы связанный аргумент был одинаковым для всех записей или чтобы разрешить им быть различными. Любое отклонение от заголовка блочной записи (MR-header) должно означать, что команда будет прервана, а запись не закодирована.

Аргумент DSFID-Lock (блокировка идентификатора DSFID) не имеет отношения к блочным записям, потому что это часть преамбулы записи, и тогда его необходимо рассматривать как признак блокировки или отсутствия блокировки.

Команда Write-Objects (записать объекты) имеет следующие аргументы:

- Add-Objects-List (список добавленных объектов) (см. 11.1);
- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2);
- DSFID-Lock (блокировка идентификатора DSFID) (см. 0 7.4.15);
- Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) (см. 11.4);
- Multiple-Records-Constructs (конструкции блочных записей) (см. 11.8);
- Packed-Object-Constructs (конструкции упакованного объекта) (см. 11.12);
- Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток) (см. 7.4.60);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Write-Objects (записать объекты)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 17
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
DSFID-Constructs-list (список конструкций идентификаторов DSFID): [OPTIONAL] List of <DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции идентификаторов DSFID>)
Ext-DSFID-Constructs-list (список конструкции расширенных идентификаторов DSFID): [OPTIONAL] List of <Ext-DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции расширенных идентификаторов DSFID>)
DSFID-Lock (блокировка идентификатора DSFID): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)
Если установлено значение TRUE (ИСТИНА), устройство опроса должно блокировать идентификатор DSFID
Add-Objects-List (список добавленных объектов): List of <Add-Objects> (список <добавленные объекты>
Packed-Object-Constructs (конструкции упакованного объекта): [OPTIONAL] List of <Packed-Object-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции упакованного объекта>)
Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток): INTEGER [OPTIONAL] (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ [ОПЦИОНАЛЬНЫЙ])
Multiple-Records-Constructs (конструкции блочных записей): [OPTIONAL] List of <Multiple-Records-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции блочных записей>)

10.11.2 Ответ на команду Write-Objects (записать объекты)

Ответ на команду Write-Objects (записать объекты) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4.);
- Write-Responses (ответы на команду записи) (см. 11.18).

Дополнительные значения Completion-Code (код завершения) применяются к каждому идентификатору объекта и включены в аргумент Write-Responses (ответы на команду записи).

Ответ на команду Write-Objects (записать объекты)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 17
Write-Responses-List (список ответов на команду «записать объекты»): List of <Write-Responses> (список <ответы на команду «записать объекты»>)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
29	Object-Not-Editable (нередактируемый объект)
31	Packed-Object-ID-Table-Not-Recognised-No-Encoding (таблица идентификатора упакованного объекта не распознана, не кодируется)
32	Tag-Data-Profile-ID-Table-Not-Recognised (таблица идентификаторов профиля данных радиочастотных меток не распознана)
33	Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)

36	Command-Cannot-Process-Monomorphic-UII (команда не может обрабатывать мономорфный идентификатор UII)
37	Data-CRC-Not-Applied (код CRC к данным не применяется)
38	Length-Not-Encoded-In-DSFID (длина не закодирована в идентификаторе DSFID)
43	Data-Format-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с форматом данных)
44	Access-Method-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с методом доступа)
45	Sector-Identifier-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с идентификатором сектора)
46	Record-Preamble-Not-Configured (преамбула записи не сконфигурирована)
47	Record-Preamble-Not-Locked (преамбула записи не заблокирована)
255	Execution-Error (ошибка выполнения)

Дополнительные Completion-Code (коды завершения) применяются к списку ответов на команду Write-Response-List (записать объекты)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.12 Считать объекты (Read-Objects)

10.12.1 Команда Read-Objects (считать объекты)

Эта команда не должна использоваться для считывания какого-либо объекта данных или другой части блочной записи. Соответствующие процедуры определены в 10.27.

Команда Read-Objects (считать объекты) дает указание устройству опроса на считывание набора из одного или нескольких идентификаторов объектов и связанных объектов из радиочастотной метки. Аргумент команды может использоваться для проверки того, что идентификатор объекта не дублируется в радиочастотной метке. В результате выполнения команды должна быть обработана только одна радиочастотная метка, чтобы гарантировать надежность процесса считывания.

Команда поддерживает аргумент, который позволяет приложению прописывать старший адрес ячейки в радиочастотной метке, после которой считывание прекращается. Этот аргумент включает в себя функции исходной команды readFirstObject (считать первый объект), но не ограничивается только одним идентификатором объекта. Этот аргумент команды поддерживает возможности радиоинтерфейса, которые могут оказаться более быстрыми, чем считывание именованного(ых) идентификатора(ов) объекта(ов). Таким образом, приложение может выбирать, чтобы наиболее часто используемый(ые) объект(ы) сохранялся (сохранялись) в радиочастотной метке первым(и). Необходимо определить значение (в байтах) для аргумента Max-App-Length (максимальная длина для приложения), которое может быть достигнуто имитацией кодирования или с помощью короткого тестового периода, изменяя значение этого аргумента до тех пор, пока команда не достигнет высокой вероятности считывания запрошенного(ых) идентификатора(ов) объекта(ов) и объекта(ов).

Эта команда может использоваться для считывания мономорфного идентификатора UII путем объявления соответствующего идентификатора объекта как единственной позиции в списке считанных объектов (Read-Objects) в сочетании с четвертым значением аргумента Read-Type (тип считывания). Процессор данных (data processor) проверяет, зарегистрирован ли идентификатор объекта как часть позиции мономорфного идентификатора UII в регистре конструкций данных по ИСО/МЭК 15961-2*.

Если это так, то закодированные байты переводятся из уплотненного состояния в правила, определенные в регистре, и включаются в ответ.

Если нет, то возвращается соответствующая ошибка.

Команда Read-Objects (считать объекты) имеет следующие аргументы:

- Max-App-Length (максимальная длина для приложения) (см. 7.4.32);
- Read-Objects (считать объекты) (см. 11.13);
- Read-Type (тип считывания) (см. 7.4.53);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

* *См.* [2].

Команда Read-Objects (считать объекты)
Module-OID (модуль идентификатора объекта): ИДЕНТИФИКАТОР ОБЪЕКТА (OBJECT IDENTIFIER) = = 1 0 15961 126 18
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Read-Type (тип считывания): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	Read-1st-Objects (считывание первых объектов)
1	Read-Multiple-Objects (считывание блочных объектов)
2	Read-All-Objects (считывание всех объектов)
3	Read-Monomorphic-UII (считывание мономорфных идентификаторов UII)

Max-App-Length (максимальная длина приложения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (1..65535)
Это относится только к Read-Type (тип считывания) (0) и выражается в байтах
Read-Objects-List (список считанных объектов): List of <Read-Objects> (список <считанные объекты>)
Это не относится к Read-All-Objects (считывание всех объектов) (2)

Если аргумент Check-Duplicate (проверка дубликата) установлен в значение FALSE (ЛОЖЬ) в аргументе Read-Object-List (список считанных объектов), то, найдя запрошенный идентификатор объекта, устройство опроса возвращает первый объект, обнаруженный без проверки на наличие дубликатов. Если для аргумента Check-Duplicate (проверка дубликата) установлено значение TRUE (ИСТИНА), устройство опроса проверяет наличие дубликатов идентификаторов объектов. Если обнаружено более одного экземпляра запрашиваемого идентификатора объекта, устройство опроса должно вернуть первый обнаруженный объект и указать наличие дубликатов с соответствующим значением кода завершения (Completion-Code).

10.12.2 Ответ на команду Read-Objects (считать объекты)

Ответ на команду Read-Objects (считать объекты) имеет следующие аргументы:
- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- Read-Objects-Response-List (список ответов на считывание объектов) (см. 11.14).

Ответ на команду Read-Objects (считать объекты)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 18
Read-Objects-Response-List (список ответов на считывание объектов): List of <Read-Objects-Response> (список <ответы считанных объектов>)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
255	Execution-Error (ошибка выполнения)

Дополнительные Completion-Code (коды завершения) применяются к Read-Objects-Response-List (список ответов на считывание объектов)
Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.13 Записать объекты в сегментированную память радиочастотной метки (Write-Objects-Segmented-Memory-Tag)

10.13.1 Команда Write-Objects-Segmented-Memory-Tag (записать объекты в сегментированную память радиочастотной метки)

Команда Write-Objects-Segmented-Memory-Tag (записать объекты в сегментированную память радиочастотной метки) аналогична команде Write-Objects (записать объекты), за исключением того, что

она предназначена для записи данных в выбранный банк памяти в радиочастотной метке с сегментированной памятью. Команда может быть реализована для записи исходных данных в радиочастотную метку или для добавления данных в радиочастотную метку. Команда поддерживается составным аргументом Add-Objects-List (список добавленных объектов).

Эта команда не используется для записи мономорфного идентификатора UII. Если идентификатор AFI в радиочастотной метке объявляет, что он зарегистрирован для мономорфного идентификатора UII, возвращается соответствующая ошибка и процесс кодирования прерывается. Надлежащая команда для использования определена в 10.24.

Пароль доступа (Access-Password) в команде используется для сопоставления с паролем доступа в радиочастотной метке, чтобы разрешить запись данных в радиочастотную метку.

Список конструкций идентификаторов DSFID (DSFID-Constructs-list) используется для указания метода доступа (Access-Method) и формата данных (Data-Format). Список конструкций расширенного идентификатора DSFID (Ext-DSFID-Constructs-list) используется для установки индикаторов расширенного идентификатора DSFID (Extended-DSFID) и дает указание процессору данных (data processor) на выполнение определенных процедур, например для применения к данным CRC (избыточного циклического кода).

Аргумент DSFID-Lock (блокировка идентификатора DSFID), если он установлен, применяется ко всей строке байтов идентификатора DSFID и расширенного идентификатора DSFID.

Идентификаторы AFI и DSFID предоставляются для использования в качестве аргументов одним из следующих способов:

- если данные записываются в радиочастотную метку с пустой сегментированной памятью, то данные аргументы предоставляются как часть команды для минимизации коммуникаций;
- в зависимости от соответствующего банка памяти, если данные добавляются в радиочастотную метку с сегментированной памятью, то идентификаторы AFI и DSFID в команде должны соответствовать идентификаторам AFI и DSFID, уже закодированным в радиочастотной метке, иначе возникает ошибка, и процесс кодирования может прекратиться до того, как будут обработаны значительные объемы данных;
- кроме того, должны быть обработаны все требования, объявленные аргументами в списке конструкций расширенного идентификатора DSFID (Ext-DSFID-Constructs-list).

Если методом доступа (Access-Method) является Packed-Objects (упакованные объекты), все объекты, указанные в качестве аргументов, должны быть включены в один и тот же новый упакованный объект, добавленный после любых существующих упакованных объектов в памяти. Ряд аргументов применим только к Packed-Objects (упакованные объекты), и они определены в аргументе Packed-Object-Constructs argument (конструкции упакованного объекта).

Если методом доступа (Access-Method) является Tag-Data-Profile (профиль данных радиочастотных меток), то все объекты, указанные в качестве аргументов, должны быть включены в один и тот же Tag-Data-Profile (профиль данных радиочастотных меток). Добавление объекта, не заданного аргументом Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток), следует рассматривать как ошибку, и никакого кодирования не происходит.

Блочные записи могут быть закодированы только в банке памяти 11. Индивидуальные записи записываются только после создания заголовка блочной записи (MR-header). Детали, определенные в 10.11.1 для блочных записей, должны применяться при построении команды.

Команда Write-Objects-Segmented-Memory-Tag (записать объекты в сегментированную память радиочастотной метки) имеет следующие аргументы:

- Access-Password (пароль доступа) (см. 7.4.1);
- Add-Objects-List (список добавленных объектов) (см. 11.1);
- AFI (идентификатор AFI) (см. 7.2.2);
- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2);
- DSFID-Lock (блокировка идентификатора DSFID) (см. 7.4.15);
- Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) (см. 11.4);
- Memory-Bank (банк памяти) (см. 7.4.33);
- Multiple-Records-Constructs (конструкции блочных записей) (см. 11.8);
- Packed-Object-Constructs (конструкции упакованного объекта) (см. 11.12);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1);
- Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток) (см. 7.4.60).

Команда Write-Objects-Segmented-Memory-Tag (записать объекты в сегментированную память радиочастотной метки)

Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 126 19
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Memory-Bank (банк памяти): BIT STRING (ДВОИЧНАЯ СТРОКА)

Возможные значения:

Значение	Определение
01_2	UII-Memory (память идентификатора UII)
11_2	User-Memory (пользовательская память)

Access-Password (пароль доступа): [OPTIONAL] BYTE STRING ([ОПЦИОНАЛЬНЫЙ)] СТРОКА БАЙТОВ) (4 байта)
AFI (идентификатор AFI): БАЙТ (BYTE) [применяется только, если Memory-Bank (Банк памяти) = 01]
DSFID-Constructs-list (список конструкций идентификатора DSFID): List of <DSFID-Constructs> (список <конструкции идентификатора DSFID>)
Ext-DSFID-Constructs-list (список конструкций расширенного идентификатора DSFID): [Optional] List of <Ext-DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции расширенного идентификатора DSFID>)
DSFID-Lock (блокировка идентификатора DSFID): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), устройство опроса должно блокировать идентификатор DSFID

Add-Objects-List (список добавленных объектов): List of <Add-Objects> (список <добавленные объекты>)
Packed-Object-Constructs (конструкции упакованного объекта): [OPTIONAL] List of <Packed-Object-Constructs> ([ОПЦИОНАЛЬНЫЙ)] список <конструкции упакованных объектов>)
Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток): INTEGER [OPTIONAL] (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ [ОПЦИОНАЛЬНЫЙ])
Multiple-Records-Constructs (конструкции блочных записей): [OPTIONAL] List of <Multiple-Records-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции блочных записей>)

10.13.2 Ответ на команду Write-Objects-Segmented-Memory-Tag (записать объекты в сегментированную память радиочастотной метки)

Ответ на команду Write-Objects-Segmented-Memory-Tag (записать объекты в сегментированную память радиочастотной метки) имеет следующее наименование полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- Write-Responses (ответы на команду записи) (см. 11.18).

Дополнительные значения Completion-Codes (код завершения) применяются к каждому идентификатору объекта и включены в аргумент Write-Responses (ответы на команду записи»).

Ответ на команду Write-Objects-Segmented-Memory-Tag (записать объекты в сегментированную память радиочастотной метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 19
Write-Response-List (список ответов на команду записи): List of <Write-Responses> (список <ответы на команду записи>)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
25	Password-Mismatch (несоответствие пароля)
26	AFI-Mismatch (несоответствие идентификатора AFI)
27	DSFID-Mismatch (несоответствие идентификатора DSFID)
33	Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)

36	Command-Cannot-Process-Monomorphic-UII (команда не может обрабатывать мономорфный идентификатор UII)
37	Data-CRC-Not-Applied (код CRC к данным не применяется)
38	Length-Not-Encoded-In-DSFID (длина не закодирована в идентификаторе DSFID)
43	Data-Format-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с форматом данных)
44	Access-Method-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с методом доступа)
45	Sector-Identifier-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с идентификатором сектора)
46	Record-Preamble-Not-Configured (преамбула записи не сконфигурирована)
47	Record-Preamble-Not-Locked (преамбула записи не заблокирована)
255	Execution-Error (ошибка выполнения)

Дополнительные коды завершения (Completion-Code) применяются к списку ответов на команду записи (Write-Response-List)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.14 Записать EPC-UII (Write-EPC-UII)

10.14.1 Команда Write-EPC-UII (записать EPC-UII)

Команда Write-EPC-UII (записать EPC-UII) дает указание устройству опроса записать кодовое значение EPC в банк 01 сегментированной памяти радиочастотной метки. Команда поддерживает кодовые значения EPC различной длины, указанные в стандарте TDS*.

Пароль доступа (Access-Password) в команде используется для сопоставления с паролем радиочастотной метки для защиты от несанкционированной записи данных в радиочастотную метку.

Эта команда может использоваться для первоначальной записи кодового значения EPC в радиочастотную метку или для перезаписи значения кода. Если новое значение кода имеет меньшую длину, устройство опроса должно убедиться, что байты, представляющие часть старого кодового значения, удалены.

Команда Write-EPC-UII (записать EPC-UII) имеет следующие аргументы:

- Access-Password (пароль доступа) (см. 7.4.1);
- EPC-Code (кодовое значение EPC) (см. 7.4.19);
- Memory-Bank-Lock (блокировка банка памяти) (см. 7.4.34);
- NSI-Bits (биты NSI) (см. 7.4.40);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Write-EPC-UII (записать EPC-UII)

Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 20

Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)

Access-Password (пароль доступа): [OPTIONAL] BYTE STRING ([ОПЦИОНАЛЬНЫЙ] СТРОКА БАЙТОВ) (4 байта)

NSI-Bits (биты NSI): BIT STRING (ДВОИЧНАЯ СТРОКА)

EPC-Code (кодовое значение EPC): BYTE STRING (СТРОКА БАЙТОВ)

Memory-Bank-Lock (блокировка банка памяти): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), банк памяти должен быть целиком заблокирован

10.14.2 Ответ на команду Write-EPC-UII (записать EPC-UII)

Ответ на команду Write-EPC-UII (записать EPC-UII) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

---

* *См.* [22].

Ответ на команду Write-EPC-UII (записать EPC-UII)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 20
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
25	Password-Mismatch (несоответствие пароля)
33	Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.15 Инвентаризировать память ISO-UII (Inventory-ISO-UIImemory)

10.15.1 Команда Inventory-ISO-UIImemory (инвентаризировать память ISO-UII)

Команда Inventory-ISO-UIImemory (инвентаризировать память ISO-UII) предназначена для возврата содержания памяти идентификатора UII сегментированной памяти радиочастотной метки, учитывая, что идентификатор объекта (Object-Identifier) кодируется для идентификатора UII, отличного от кода EPC. Ответ возвращает содержимое памяти идентификатора UII для всех радиочастотных меток, чья закодированная битовая строка соответствует аргументам команды.

Аргументы, приведенные в команде, позволяют включить маску битов (bit mask) в соответствующие команды протокола радиоинтерфейса, чтобы выбрать только те радиочастотные метки, которые соответствуют маске битов. Идентификатор AFI является обязательным аргументом, а два других аргумента расширяют битовую строку, чтобы увеличить возможность отбора.

Эта команда может использоваться для считывания мономорфного идентификатора UII путем объявления соответствующего идентификатора AFI в качестве аргумента. Процессор данных (data processor) проверяет, что идентификатор AFI зарегистрирован как часть позиции мономорфного идентификатора UII в регистре конструкций данных ИСО/МЭК 15961-2:

- если идентификатор AFI зарегистрирован, то закодированные байты разуплотняются в соответствии с правилами, определенными в регистре, и включаются в ответ;
- если нет, возвращается соответствующая ошибка.

Примечание — При вызове этой команды для инвентаризации мономорфного идентификатора UII не требуется идентификатор DSIFD.

Команда Inventory-ISO-UIImemory (инвентаризировать память ISO-UII) имеет следующие аргументы:
- Additional-App-Bits (дополнительные биты приложения) (см. 7.4.2);
- AFI (идентификатор AFI) (см. 7.2.2);
- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2).

Команда Inventory-ISO-UIImemory (инвентаризировать память ISO-UII)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 21
AFI (идентификатор AFI): БАЙТ (BYTE)
DSFID-Constructs-list (список конструкций идентификатора DSFID): [OPTIONAL] List of <DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции идентификатора DSFID>)
Additional-App-Bits (дополнительные биты приложения): BIT STRING [OPTIONAL] (ДВОИЧНАЯ СТРОКА [ОПЦИОНАЛЬНЫЙ])

10.15.2 Ответ на команду Inventory-ISO-UIImemory (инвентаризировать память ISO-UII)

Ответ на команду Inventory-ISO-UIImemory (инвентаризировать память ISO-UII) имеет следующие наименования полей:
- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- ISO-UIImemory (память идентификатора UII для ISO) (см. 11.5).

Ответ на команду Inventory-ISO-UIImemory (инвентаризировать память ISO-UII)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 127 21
ISO-UII Memory-List (реестр памяти ISO-UII): List of <ISO-UIIMemory> (реестр <память ISO-UII>)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.16 Инвентаризировать память EPC-UII (Inventory-EPC-UIImemory)

10.16.1 Команда Inventory-EPC-UIImemory (инвентаризировать память EPC-UII)

Команда Inventory-EPC-UIImemory (инвентаризировать память EPC-UII) предназначена для возврата содержимого памяти идентификатора UII (UII memory) из сегментированной памяти радиочастотной метки, учитывая, что кодируется кодовое значение EPC. Ответ возвращает содержимое памяти идентификатора UII для всех радиочастотных меток, закодированная битовая строка которых соответствует аргументам команды.

Аргументы, предоставленные в команде, позволяют включить маску битов в соответствующие команды протокола радиоинтерфейса, чтобы выбрать только те радиочастотные метки, которые соответствуют маске битов. Значение Tag-Mask (маска радиочастотной метки) состоит из битовой строки, которая должна определяться посредством ссылки на соответствующие стандарты TDS*. То же самое относится к Pointer (указатель адреса), который идентифицирует первый бит непрерывной строки в радиочастотной метке, который необходимо сопоставить с маской радиочастотной метки.

Команда Inventory-EPC-UIImemory (инвентаризировать память EPC-UII) имеет следующие аргументы:

- Length-Of-Mask (длина маски) (см. 7.4.27);
- Pointer (указатель адреса) (см. 7.4.50);
- Tag-Mask (маска радиочастотной метки) (см. 7.4.61).

Команда Inventory-EPC-UIImemory (инвентаризировать память EPC-UII)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 22
Pointer (указатель адреса): ШЕСТНАДЦАТЕРИЧНЫЙ АДРЕС (HEXADECIMAL ADDRESS)
Адрес первого (msb) бита, к которому следует применять маску радиочастотной метки (Tag-Mask).
Length-Of-Mask (длина маски): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Tag-Mask (маска радиочастотной метки): BIT STRING (ДВОИЧНАЯ СТРОКА)

10.16.2 Ответ на команду Inventory-EPC-UIImemory (инвентаризировать память EPC-UII)

Ответ на команду (инвентаризировать память EPC-UII) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- EPC-UIImemory (память EPC-UII) (см. 11.3);
- Execution-Code (код выполнения) (см. 9.4).

Inventory-EPC-UIImemory response (ответ на команду «инвентаризировать память EPC-UII»)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 22
EPC-UIImemory-List (реестр памяти EPC-UII): List of <EPC-UIImemory> (реестр <память EPC-UII>)
Completion Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

* *См.* [22].

10.17 Записать пароль в сегментированную память радиочастотной метки (Write-Password-Segmented-Memory-Tag)

10.17.1 Команда Write-Password-Segmented-Memory-Tag (записать пароль в сегментированную память радиочастотной метки)

Команда Write-Password-Segmented-Memory-Tag (записать пароль в сегментированную память радиочастотной метки) дает указание устройству опроса записать один из паролей (Passwords), определенных в команде, в соответствующую память радиочастотной метки с сегментированной памятью. Для каждой команды может быть указан только один пароль. Аргумент Password-Type (тип пароля) указывает процессору данных (data processor) тип пароля и, следовательно, его место хранения в радиочастотной метке.

Команда Write-Password-Segmented-Memory-Tag (записать пароль в сегментированную память радиочастотной метки) имеет следующие аргументы:

- Password (пароль) (см. 7.4.46);
- Password-Type (тип пароля) (см. 7.4.47);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Write-Password-Segmented-Memory-Tag (записать пароль в сегментированную память радиочастотной метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 126 23
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Password-Type (тип пароля): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	Kill-Password (удаление пароля)
1	Access-Password (пароль доступа)

Password (пароль): BYTE STRING (СТРОКА БАЙТОВ)
Для Password-Type (тип пароля) со значениями 0 и 1 длина составляет 4 бита

10.17.2 Ответ на команду Write-Password-Segmented-Memory-Tag (записать пароль в сегментированную память радиочастотной метки)

Ответ на команду Write-Password-Segmented-Memory-Tag (записать пароль в сегментированную память радиочастотной метки) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Write-Password-Segmented-Memory-Tag (записать пароль в сегментированную память радиочастотной метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 – 15961 127 23
Completion-Cod (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
26	Password-Not-Written (пароль не записан)
33	Insufficient-Tag-Memory (не хватает памяти радиочастотной метки)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.18 Считать слова из сегментированной памяти радиочастотной метки (Read-Words-Segmented-Memory-Tag)

10.18.1 Команда Read-Words-Segmented-Memory-Tag (считать слова из сегментированной памяти радиочастотной метки)

Команда Read-Words-Segmented-Memory-Tag (считать слова из сегментированной памяти радиочастотной метки) дает указание устройству опроса на считывание непрерывной последовательности слов из одного из банков памяти радиочастотной метки с сегментированной памятью. Эта команда может использоваться для извлечения закодированных байтов, которые могут не быть объектно-ориентированными, такими как уникальный идентификатор Singulation-Id или пароль. Она также может быть полезна для диагностических целей.

Команда «считать слова из сегментированной памяти радиочастотной метки» (Read-Words-Segmented-Memory-Tag) имеет следующие аргументы:

- Access-Password (пароль доступа) (см. 7.4.1);
- Memory-Bank (Банк памяти) (см. 7.4.33);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1);
- Word-Count (число слов) (см. 7.4.63);
- Word-Pointer (указатель слова) (см. 7.4.64).

Команда Read-Words-Segmented-Memory-Tag (считать слова из сегментированной памяти радиочастотной метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 24
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Memory-Bank (Банк памяти): BIT STRING (ДВОИЧНАЯ СТРОКА)
Возможные значения: (00..11)
Word-Pointer (указатель слов): ШЕСТНАДЦАТЕРИЧНЫЙ АДРЕС (HEXADECIMAL ADDRESS)
Word-Count (число слов): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Access-Password (пароль доступа): [OPTIONAL (ОПЦИОНАЛЬНЫЙ)] BYTE STRING (СТРОКА БАЙТОВ) (4 байта)

10.18.2 Ответ на команду Read-Words-Segmented-Memory-Tag (считать слова из сегментированной памяти радиочастотной метки)

Ответ на команду Read-Words-Segmented-Memory-Tag (считать слова из сегментированной памяти радиочастотной метки) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- Read-Data (считывание данных) (см. 7.5.13).

Ответ на команду Read-Words-Segmented-Memory-Tag (считать слова из сегментированной памяти радиочастотной метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 127 24
Read-Data (считывание данных): BYTE STRING (СТРОКА БАЙТОВ)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
25	Password-Mismatch (несоответствие пароля)
254	Undefined-Command-Error (неопределенная ошибка команды)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.19 Уничтожить сегментированную память радиочастотной метки (Kill-Segmented-Memory-Tag)

10.19.1 Команда Kill-Segmented-Memory-Tag (уничтожить сегментированную память радиочастотной метки)

Команда Kill-Segmented-Memory-Tag (уничтожить сегментированную память радиочастотной метки) дает указание устройству опроса на применение соответствующих протоколов радиоинтерфейса, чтобы в будущем радиочастотная метка не считывалась. Аргумент Kill-Password (уничтожение пароля) в этой команде должен соответствовать паролю, закодированному в радиочастотной метке.

Команда Kill-Segmented-Memory-Tag (уничтожить сегментированную память радиочастотной метки) имеет следующие аргументы:

- Kill-Password (уничтожение пароля) (см. 7.4.26);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Kill-Segmented-Memory-Tag (уничтожить сегментированную память радиочастотной метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 25
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Kill-Password (уничтожение пароля): BYTE STRING (СТРОКА БАЙТОВ) (4 байта)

10.19.2 Ответ на команду Kill-Segmented-Memory-Tag (уничтожить сегментированную память радиочастотной метки)

Ответ на команду Kill-Segmented-Memory-Tag (уничтожить сегментированную память радиочастотной метки) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Kill-Segmented-Memory-Tag (уничтожить сегментированную память радиочастотной метки)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 127 25
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
27	Zero-Kill-Password-Error (нулевая ошибка уничтожения пароля)
28	Kill-Failed (уничтожение не выполнено)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.20 Удалить упакованный объект (Delete-Packed-Object)

10.20.1 Команда Delete-Packed-Object (удалить упакованный объект)

Команда Delete-Packed-Object (удалить упакованный объект) дает указание устройству опроса на удаление упакованного объекта (Packed-Object), в котором содержится указанный идентификатор объекта (Object-Identifier). Идентификатор объекта просто выступает в качестве псевдонима для идентификации определенного упакованного объекта (Packed-Object). Для каждой команды необходимо запрограммировать только одну радиочастотную метку и только один идентификатор объекта, чтобы гарантировать надежность процесса удаления. Функция удаления требует удаления связанного упакованного объекта из радиочастотной метки и замены его заполняющими байтами.

Эта процедура может не выполниться, если упакованный объект заблокирован. В случае, если это будет обнаружено, в ответе будет возвращен соответствующий Completion-Code (код завершения).

Если для флага Check-Duplicate (проверка дубликата) установлено значение TRUE (ИСТИНА), устройство опроса перед удалением запрошенного упакованного объекта проверяет, что в радиочастотной метке имеется только один экземпляр запрашиваемого идентификатора объекта. Если устройство опроса обнаруживает, что радиочастотная метка кодирует более одного экземпляра ссылочно-

го идентификатора объекта, оно не должно выполнять функцию «удаление упакованного объекта» и должно вернуть соответствующий Completion-Code (код завершения).

Если для флага Check-Duplicate (проверка дубликата) установлено значение FALSE (ЛОЖЬ), устройство опроса должно удалить первое вхождение упакованного объекта, которое содержит указанный идентификатор объекта.

Примечание — Это аргумент, который фактически не обеспечивает защиту от дублирующих идентификаторов объекта. Его следует использовать только тогда, когда существует большая вероятность отсутствия дубликатов.

Команда Delete-Packed-Object (удалить упакованный объект) имеет следующие аргументы:

- Check-Duplicate (проверка дубликата) (см. 7.4.10);
- Object-Identifier (идентификатор объекта) (см. 7.3.3);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Delete-Packed-Object (удалить упакованный объект)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 26
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)
Check-Duplicate (проверка дубликата): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)
Если установлено значение TRUE (ИСТИНА), устройство опроса проверяет наличие только одного вхождения идентификатора объекта в радиочастотную метку

10.20.2 Ответ на команду Delete-Packed-Object (удалить упакованный объект)

Ответ на команду Delete-Packed-Object (удалить упакованный объект) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Delete-Packed-Object (удалить упакованный объект)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 127 26
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
10	Duplicate-Object (дубликат объекта)
12	Object-Not-Deleted (объект не удален)
13	Object-Identifier-Not-Found (идентификатор объекта не обнаружен)
14	Object-Locked-Could-Not-Delete (блокированный объект не может быть удален)
37	Data-CRC-Not-Applied (код CRC к данным не применяется)
38	Length-Not-Encoded-In-DSFID (длина не закодирована в идентификаторе DSFID)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.21 Изменить структуру упакованного объекта (Modify-Packed-Object-Structure)

10.21.1 Команда Modify-Packed-Object-Structure (изменить структуру упакованного объекта)

Команда Modify-Packed-Object-Structure (изменить структуру упакованного объекта) используется для изменения структуры упакованного объекта (Packed-Object). Структура упакованного объекта может быть изменена для определения конкретного типа каталога, если упакованный объект был создан с использованием типа указателя упакованного объекта (Packed-Object pointer), как указано в 7.4.48. Команда определяет, какой тип каталога должен быть применен. Идентификатор объекта

(Object-Identifier) просто выступает в качестве псевдонима для идентификации определенного упакованного объекта. Для каждой команды необходимо запрограммировать только одну радиочастотную метку и только один идентификатор объекта, чтобы гарантировать надежность процесса модификации.

Эта процедура может быть невыполнимой, если упакованный объект заблокирован. В случае, если блокировка обнаружится, в ответ вернется соответствующий Completion-Code (код завершения).

В аргументе Packed-Object-Constructs (конструкции упакованного объекта) определен ряд аргументов. Эти аргументы должны использоваться, если они включены, но только в том случае, если они имеют отношение к типу упакованного объекта, который изменяется командой. Если они не относятся к текущему упакованному объекту, параметры должны игнорироваться.

Если у упакованного объекта уже определен конкретный тип каталога, то будет возвращен соответствующий Completion-Code (код завершения).

Если для флага Check-Duplicate (проверка дубликата) установлено значение TRUE (ИСТИНА), то перед тем, как изменить запрошенный упакованный объект, устройство опроса проверит, что в радиочастотной метке имеется только один экземпляр запрашиваемого идентификатора объекта. Если устройство опроса обнаружит, что радиочастотная метка кодирует более одного экземпляра базового идентификатора объекта, оно не должно выполнять функцию Modify-Packed-Object-Structure (изменить структуру упакованного объекта) и должно вернуть соответствующий Completion-Code (код завершения).

Если для флага Check-Duplicate (проверка дубликата) установлено значение FALSE (ЛОЖЬ), устройство опроса должно изменить первое вхождение упакованного объекта, которое содержит указанный идентификатор объекта.

Это аргумент, который фактически не обеспечивает защиту от дублирующих идентификаторов объекта. Его следует использовать только тогда, когда существует большая вероятность отсутствия дубликатов.

Команда Modify-Packed-Object-Structure (изменить структуру упакованного объекта) имеет следующие аргументы:

- Check-Duplicate (проверка дубликата) (см. 7.4.10);
- Object-Identifier (идентификатор объекта) (см. 7.3.3);
- Packed-Object-Constructs (конструкции упакованного объекта) (см. 11.12);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Modify-Packed-Object-Structure (изменить структуру упакованного объекта)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 27
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0…255)
Object-Identifier (идентификатор объекта): ИДЕНТИФИКАТОР ОБЪЕКТА
Check-Duplicate (проверка дубликата): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)
Если установлено значение TRUE (ИСТИНА), устройство опроса должно проверить, что есть только одно вхождение идентификатора объекта
Packed-Object-Constructs-List (список конструкций упакованного объекта): [OPTIONAL] List of <Packed Object Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции упакованного объекта>)

10.21.2 Ответ на команду Modify-Packed-Object-Structure (изменить структуру упакованного объекта)

Ответ на команду Modify-Packed-Object-Structure (изменить структуру упакованного объекта) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Modify-Packed-Object-Structure (изменить структуру упакованного объекта)
Module-OID (Модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 27
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
7	Object-Locked-Could-Not-Modify (блокированный объект не может быть изменен)

8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
10	Duplicate-Object (дубликат объекта)
13	Object-Identifier-Not-Found (идентификатор объекта не обнаружен)
30	Directory-Already-Defined (каталог уже определен)
33	Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)
37	Data-CRC-Not-Applied (код CRC к данным не применяется)
38	Length-Not-Encoded-In-DSFID (длина не закодирована в идентификаторе DSFID)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.22 Записать сегменты радиочастотной метки типа 6TypeD (Write-Segments-6TypeD-Tag)

10.22.1 Команда Write-Segments-6TypeD-Tag (записать сегменты радиочастотной метки типа 6TypeD)

Команда Write-Segments-6TypeD-Tag (записать сегменты радиочастотной метки типа 6TypeD) дает указание процессору данных (data processor) в ИСО/МЭК 15962 на запись данных в сегмент идентификатора UII (UII segment), в сегмент, связанный с предметом (item-related segment), или в оба сегмента. Команда может быть реализована для записи исходных данных в радиочастотную метку или для добавления данных в радиочастотную метку. Команда поддерживается составным аргументом Add Objects-List (список добавленных объектов), который применяется к сегменту, связанному с предметом.

Эта команда не должна использоваться для записи мономорфного идентификатора UII. Если идентификатор AFI в радиочастотной метке объявляет, что он зарегистрирован для мономорфного идентификатора UII, возвращается соответствующая ошибка и процесс кодирования прерывается. Надлежащая команда для использования определена в 10.24.

Если данные сначала закодированы в радиочастотной метке, то идентификатор AFI, идентификатор DSFID-UII и идентификатор DSFID, связанный с предметом (Item-Related-DSFID), если этот сегмент кодируется, должны быть включены в закодированную строку байтов. Если какой-либо из этих трех идентификаторов уже закодирован в радиочастотной метке и есть несовпадение с эквивалентным аргументом в команде, тогда процедура должна быть прервана.

Если методом доступа (Access-Method) является Packed-Objects (упакованные объекты), все объекты, указанные в качестве аргументов, должны быть включены в один и тот же новый упакованный объект, добавленный после любых существующих упакованных объектов в памяти. Ряд аргументов применим только к упакованным объектам, и они определены в аргументе Packed-Object-Constructs (конструкции упакованного объекта).

Если методом доступа (Access-Method) является Tag-Data-Profile (профиль данных радиочастотных меток), то все объекты, указанные в качестве аргументов, должны быть включены в один и тот же Tag-Data-Profile (профиль данных радиочастотных меток). Добавление объекта, не заданного аргументом Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток), следует рассматривать как ошибку, и никакого кодирования не происходит.

Блочные записи могут быть закодированы только в связанном с предметом (item-related) сегменте. Отдельные записи записываются только после создания заголовка блочной записи (MR-header). Детали, определенные в 10.11.1, должны применяться при построении команды.

Команда Write-Segments-6TypeD-Tag (записать сегменты радиочастотной метки типа 6TypeD) имеет следующие аргументы:

- AFI (идентификатор AFI) (см. 7.2.2);
- Item-Related-Add-Objects-List (список добавленных объектов, связанных с предметами) (см. 11.6);
- Item-Related-DSFID-Constructs (конструкции идентификатора DSFID, связанного с предметом) (см. 11.7);
- Lock-UII-Segment-Arguments (аргументы блокировки сегмента идентификатора UII) (см. 7.4.31);
- Memory-Segment (сегмент памяти) (см. 7.4.36);
- Multiple-Records-Constructs (конструкции блочных записей) (см. 11.8);
- Packed-Object-Constructs (конструкции упакованного объекта) (см. 11.12);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1);

- Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток) (см. 7.4.60);
- UII-Add-Objects-List (список добавленных объектов в формате идентификатора UII) (см. 11.16);
- UII-DSFID-Constructs (конструкции UII-DSFID) (см. 11.17).

Команда Write-Segments-6TypeD-Tag (записать сегменты радиочастотной метки типа 6TypeD)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 28
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0.255)
Memory-Segment (сегмент памяти): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
1	UII segment (сегмент идентификатора UII)
2	Item-related data segment (сегмент данных, связанный с предметом)
3	Both segments presented as objects (оба сегмента представлены как объекты)

AFI (идентификатор AFI): BYTE (БАЙТ)

Не требуется для значения Memory-Segment (сегмент памяти) = 2 (сегмент данных, связанный с предметом)

UII-DSFID-Constructs-list (список конструкций UII-DSFID): [OPTIONAL] List of <DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции идентификаторов DSFID>)

Не рекомендуется для значения Memory-Segment (сегмент памяти) = 2 (сегмент данных, связанный с предметом)

UII-Add-Objects-List (список добавленных объектов в формате идентификатора UII): List of <Add-Objects> (список <добавленные объекты>)

Требуется только для значения Memory-Segment (сегмент памяти) = 1 (сегмент идентификатора UII), должен содержать один идентификатор объекта и объект

Lock-UII-Segment-Arguments (аргументы блокировки сегмента идентификатора UII): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Этот аргумент имеет приоритет над аргументом блокировки в любом месте команды

Возможные значения:

Значение	Определение
1	Protocol Control word (слово управления протоколом)
2	DSFID (идентификатор DSFID)
3	PC word + DSFID (слово управления протоколом + идентификатор DSFID)
4	UII Data-Set (набор данных идентификатора UII), но не слово управления протоколом (PC word), не идентификатор DSFID
5	PC word + UII Data-Set (слово управления протоколом + набор данных идентификатора UII) (данная комбинация применима к мономорфному идентификатору UII)
6	DSFID + UII Data-Set (идентификатор DSFID + набор данных идентификатора UII)
7	complete UII segment (полный сегмент идентификатора UII), включая код CRC-16

Item-Related-DSFID-Constructs-list (список конструкций DSFID, связанных с предметами): [OPTIONAL] List of <DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции идентификатора DSFID>)

Не требуется для значения Memory-Segment (сегмент памяти) = 1 (сегмент идентификатора UII)

Item-Related-Add-Objects-List (список добавленных объектов, связанных с предметами): List of <Add-Objects> (список <добавленные объекты>)

Не требуется для значения Memory-Segment (сегмент памяти) = 1 (сегмент идентификатора UII)

Packed-Object-Constructs (конструкции упакованного объекта): [OPTIONAL] List of <Packed-Object-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции упакованного объекта>)

Не требуется для значения Memory-Segment (сегмент памяти) = 1 (сегмент идентификатора UII)

Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток): INTEGER [OPTIONAL] (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ [ОПЦИОНАЛЬНЫЙ])

Не требуется для значения Memory-Segment (сегмент памяти) = 1 (сегмент идентификатора UII)

Multiple-Records-Constructs (конструкции блочных записей): [OPTIONAL] List of <Multiple-Records-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции блочных записей>)

10.22.2 Ответ на команду Write-Segments-6TypeD-Tag (записать сегменты радиочастотной метки типа 6TypeD)

Ответ на команду Write-Segments-6TypeD-Tag (записать сегменты радиочастотной метки типа 6TypeD) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Write-Segments-6TypeD-Tag (записать сегменты радиочастотной метки типа 6TypeD)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 – 15961 127 28
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
26	AFI-Mismatch (несоответствие идентификатора AFI)
27	DSFID-Mismatch (несоответствие идентификатора DSFID)
31	Packed-Object-ID-Table-Not-Recognised-No-Encoding (Таблица идентификаторов упакованного объекта не распознана, кодирование не выполнено)
32	Tag-Data-Profile-ID-Table (таблица идентификаторов профиля данных радиочастотных меток не распознана)
33	Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)
36	Command-Cannot-Process-Monomorphic-UII (команда не может обрабатывать мономорфный идентификатор UII)
43	Data-Format-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с форматом данных)
44	Access-Method-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с методом доступа)
45	Sector-Identifier-Not-Compatible-Multiple-Records-Header (заголовок блочной записи несовместим с идентификатором сектора)
46	Record-Preamble-Not-Configured (преамбула записи не сконфигурирована)
47	Record-Preamble-Not-Locked (преамбула записи не заблокирована)
255	Execution-Error (ошибка выполнения)

Дополнительные Completion-Codes (коды завершения) применяются к Write-Response-List (список ответов на запись)
Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.23 Считать сегменты радиочастотной метки типа 6TypeD (Read-Segments-6TypeD-Tag)

10.23.1 Команда Read-Segments-6TypeD-Tag (считать сегменты радиочастотной метки типа 6TypeD)

Эта команда не должна использоваться для считывания какого-либо объекта данных или иной части блочной записи. Соответствующие процедуры определены в 10.27.

Команда Read-Segments-6TypeD-Tag (считать сегменты радиочастотной метки типа 6TypeD) дает указание устройству опроса на считывание непрерывной последовательности слов из всех сегментов памяти радиочастотной метки типа 6TypeD, а затем на подразделение на сегменты и другие составные части. Эта команда также может быть полезна для диагностических целей.

Поскольку протокол радиоинтерфейса способен доставлять все полезное содержимое радиочастотной метки типа 6TypeD, приложение может вызывать идентификатор(ы) объекта (Object-Identifier) и объект(ы) (Object) из сегмента идентификатора UII и/или связанный с предметом сегмент данных. Основное предположение этой команды состоит в том, что в памяти идентификатора UII закодирован только один идентификатор объекта и набор данных объекта.

Эта команда может использоваться для считывания мономорфного идентификатора UII путем объявления соответствующего идентификатора AFI в качестве аргумента. Процессор данных (data processor) проверяет, зарегистрирован ли идентификатор AFI как часть позиции мономорфного идентификатора UII в регистре конструкций данных по ИСО/МЭК 15961-2:

- если это так, закодированные байты разуплотняются в соответствии с правилами, определенными в регистре, и включаются в ответ.
- если нет, возвращается соответствующая ошибка.

Примечание — Идентификатор UII-DSFID не требуется при вызове этой команды для считывания мономорфного идентификатора UII.

Аргумент Read-Type (тип считывания) применяется только в том случае, если объекты (Objects) должны быть возвращены из сегмента, связанного с предметом. Аргумент позволяет приложению запрашивать выбранный(ые) и объявленный(ые) идентификатор(ы) объекта (Object-Identifier), который(ые) должен (должны) быть обработан(ы), или запросить обработку идентификатора (всех идентификаторов) объекта.

Можно, установив значение Memory-Segment (сегмент памяти) = 4, вызвать все байты, закодированные в сегменте идентификатора UII и сегменте, связанном с предметом, а по сути, байтовую строку, определенную в ИСО/МЭК18000-64, как выходные данные устройства опроса. Поскольку это обходит любой процесс декодирования процессором данных (data processor) по ИСО/МЭК 15962, байтовая строка в ответе может использоваться для диагностических целей исходного кодирования.

Команда Read-Segments-6TypeD-Tag (считать сегменты радиочастотной метки типа 6TypeD) имеет следующие аргументы:

- AFI (идентификатор AFI) (см. 7.2.2);
- Item-Related-DSFID-Constructs (конструкции идентификатора DSFID, связанного с предметом) (см. 11.7);
- Memory-Segment (сегмент памяти) (см. 7.4.36);
- Read-Objects (считывание объектов) (см. 11.13);
- Read-Type (тип считывания) (см. 7.4.53);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1);
- UII-DSFID-Constructs (конструкции UII-DSFID) (см. 11.17).

Команда Read-Segments-6TypeD-Tag (считать сегменты радиочастотной метки типа 6TypeD)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 29
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Memory-Segment (сегмент памяти): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
1	UII segment (сегмент идентификатора UII)
2	Item-related data segment (сегмент данных, связанный с предметом)
3	Оба сегмента представлены как объекты
4	Оба сегмента как неинтерпретированные байты

AFI (идентификатор AFI): BYTE (БАЙТ)
UII-DSFID-Constructs-list (список конструкций UII-DSFID): [OPTIONAL] List of <DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции идентификатора DSFID>)

Не требуется для значения Memory-Segment (сегмент памяти) = 2 (сегмент данных, связанный с предметом)

Не требуется, если идентификатор AFI предназначен для мономорфного идентификатора UII
Item-Related-DSFID-Constructs-list (список конструкций идентификатора DSFID, связанного с предметом): [OPTIONAL] List of <DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции DSFID>)

Не требуется для значения Memory-Segment (сегмент памяти) = 1 (сегмент UII)
Read-Type (тип считывания): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Не требуется для значения Memory-Segment (сегмент памяти) = 1 (сегмент идентификатора UII)

Возможные значения:

Значение	Определение
1	Read-Multiple-Objects (считывание блочных объектов)
2	Read-All-Objects (считывание всех объектов)

Read-Objects-List (список считанных объектов): List of <Read-Objects> (список <считанные объекты>)

Не используется для Read-All-Objects (считывание всех объектов) (2)

10.23.2 Ответ на команду Read-Segments-6TypeD-Tag (считать сегменты радиочастотной метки типа 6TypeD)

Ответ на команду Read-Segments-6TypeD-Tag (считать сегменты радиочастотной метки типа 6TypeD) имеет следующие аргументы:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- ISO-UIImemory (память ISO-UII) (см. 11.5);
- Length-Lock Byte (байт блокировки длины) (см. 7.5.4);
- Read-Data (считывание данных) (см. 7.5.13);
- Read-Objects-Response-List (список ответов на считывание объектов) (см. 11.14).

Ответ на команду Read-Segments-6TypeD-Tag (считать сегменты радиочастотной метки типа 6TypeD)

Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 127 29

ISO-UII Memory-List (список памяти ISO-UII): List of <ISO-UIIMemory> (список <память ISO UII>)

Этот аргумент ответа включается, когда запрашивается содержимое сегмента идентификатора UII

Length-Lock Byte (байт блокировки длины): BYTE (БАЙТ)

Этот аргумент ответа включается, когда запрашивается содержимое сегмента данных, связанного с предметом.

Read-Objects-Response-List (список ответов на считывание объектов): List of <Read-Objects-Response> (список <ответы считанных объектов>)

Этот аргумент ответа включается, когда запрашивается содержимое сегмента данных, связанного с предметом

Read-Data (считывание данных): BYTE STRING (СТРОКА БАЙТОВ)

Этот аргумент ответа включается, когда значение Memory-Segment (сегмент памяти) = 4 (оба сегмента как неинтерпретированные байты)

Completion Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.24 Записать мономорфный идентификатор UII (Write-Monomorphic-UII)

10.24.1 Команда Write-Monomorphic-UII (записать мономорфный идентификатор UII)

Команда Write-Monomorphic-UII (записать мономорфный идентификатор UII) дает указание процессору данных (data processor) по ИСО/МЭК 15962 на запись мономорфного идентификатора UII либо изменение существующего мономорфного идентификатора UII. Аргументы применяются выборочно в зависимости от типа адресуемой радиочастотной метки.

Мономорфный идентификатор UII поддерживается только закодированным идентификатором AFI, а идентификатор DSFID не кодируется. Идентификатор объекта (Object-Identifier) требуется только для связи между приложением и процессором данных (data processor). Он не закодирован в радиочастотной метке. В команде должен использоваться только параметр уплотнения (Compact-Parameter) со значением 5 (мономорфный идентификатор UII).

Общий процесс кодирования требует, чтобы процессор данных (data processor) проверял, соответствует ли идентификатор AFI в команде идентификатору AFI для мономорфного идентификатора UII в конструкции регистра конструкций данных по ИСО/МЭК 15961-2*. Если совпадение невозможно либо потому, что идентификатор AFI не зарегистрирован для мономорфного идентификатора UII, либо потому, что в регистре не обнаружено совпадающего идентификатора AFI, процесс кодирования прерывается.

Если идентификатор AFI соответствует, идентификатор объекта также проверяется на соответствие с регистром конструкций данных по ИСО/МЭК 15961-2*. Несоответствие генерирует соответствующий Completion-Code (код завершения), но это следует рассматривать только как предупреждение о создании команды. Процесс продолжается, потому что идентификатор объекта не закодирован. Процессор данных (data processor) использует явным образом определенную схему уплотнения, связанную с идентификатором AFI в регистре конструкций данных по ИСО/МЭК 15961-2* для кодирования мономорфного идентификатора UII.

Если аргумент Memory-Type (тип памяти) равен 0 (банк памяти MB01 радиочастотной метки типа C), первым требованием является считывание содержимого банка памяти MB01, чтобы задать любое существующее кодирование. Если он заблокирован, процесс прерывается, в противном случае процесс продолжается. Процесс кодирования уплотняет объект, и, если это приводит к нечетному числу байтов, добавляется байт-ограничитель 00_{16}. Строка протокола контролируется, включая идентификатор AFI и биты длины. Если пароль доступа (Access-Password) находится в команде, он используется для сопоставления с радиочастотной меткой для защиты от несанкционированной записи данных в радиочастотную метку. Если эта команда используется для перезаписи банка памяти MB01 с новым мономорфным идентификатором UII, необходимо сравнить длину текущей и новой строки байтов. Если новое кодовое значение имеет более короткую длину, устройству опроса необходимо убедиться, что байты, представляющие часть старого кодового значения, перезаписаны нулевыми байтами.

Если аргумент Memory-Type (тип памяти) равен 1 (сегмент идентификатора UII радиочастотной метки типа D), первым требованием является считывание содержимого сегмента идентификатора UII для установки любого существующего кодирования. Если он заблокирован, процесс прерывается, в противном случае процесс продолжается. Процесс кодирования уплотняет объект, и, если это приводит к нечетному числу байтов, добавляется байт-ограничитель 00_{16}. Строка протокола контролируется, включая идентификатор AFI и биты длины. Если эта команда используется для перезаписи сегмента идентификатора UII с помощью нового мономорфного идентификатора UII, необходимо сравнить длину текущей и новой строки байтов. Если новое кодовое значение имеет другую длину (т. е. короче или длиннее) по сравнению с существующим мономорфным идентификатором UII, устройство считывания должно проверить, закодирован ли сегмент, связанный с предметом. Процедура перезаписи, как определено в ИСО/МЭК 15962, должна учитывать, имеются ли данные, относящиеся к предметам, в системе радиочастотной идентификации и заблокированы ли какие-либо из них. Это необходимо, потому что кодирование в радиочастотной метке типа D является непрерывным между сегментами.

Если аргумент Memory-Type (тип памяти) равен 2 [метка с единственной памятью (single memory tag)], первым требованием является считывание идентификатора AFI в радиочастотной метке (которое может быть закодировано в отдельной области памяти) и не менее 16 байт из содержимого пользовательской памяти, чтобы задать любое существующее кодирование. Если закодированные байты обнаружены, то процесс продолжается до тех пор, пока не будет обнаружена строка из четырех нулевых байтов. Если идентификатор AFI или любая часть пользовательской памяти заблокирована, процесс прерывается, в противном случае процесс продолжается. Процесс кодирования уплотняет объект и добавляет длину уплотненного мономорфного идентификатора UII в качестве префикса. Если эта команда используется для перезаписи с новым мономорфным идентификатором UII, необходимо сравнить длину текущей и новой строк байтов. Если новое кодое значение имеет более короткую длину, устройству опроса необходимо убедиться, что байты, представляющие часть старого кодового значения, перезаписаны нулевыми байтами. Если надлежащий идентификатор AFI для мономорфного идентификатора UII еще не закодирован в радиочастотной метке, то он кодируется.

Команда Write-Monomorphic-UII (записать мономорфный идентификатор UII) имеет следующие аргументы:

- Access-Password (пароль доступа) (см. 7.4.1);
- AFI (идентификатор AFI) (см. 7.2.2);
- AFI-Lock (блокировка идентификатора AFI) (см. 7.4.3);

---

* *См.* [*9*].

- Compact-Parameter (параметр уплотнения) (см. 7.3.6);
- Memory-Bank-Lock (блокировка банка памяти) (см. 7.4.34);
- Memory-Type (тип памяти) (см. 7.4.37);
- Object (объект) (см. 7.3.5);
- Object-Identifier (идентификатор объекта) (см. 7.3.3);
- Object-Lock (блокировка объекта) (см. 7.3.7);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Write-Monomorphic-UII (записать мономорфный идентификатор UII)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 30
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Memory-Type (тип памяти): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (0..15)

Возможные значения:

Значение	Определение
0	Type C MB01 (банк памяти MB01 радиочастотной метки типа C)
1	Type D UII segment (сегмент идентификатора UII радиочастотной метки типа D)
2	single memory tag (радиочастотная метка с единственной памятью)

Access-Passwor (пароль доступа): [OPTIONAL BYTE STRING] ([ОПЦИОНАЛЬНЫЙ] СТРОКА БАЙТОВ) (4 байта)

Может применяться только для Memory-Type (тип памяти): 0 [Type C MB01 (банк памяти MB01 радиочастотной метки типа C)]. Кроме того, он не является обязательным
AFI (идентификатор AFI): BYTE (БАЙТ)
AFI-Lock (блокировка идентификатора AFI): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Может быть применена только для Memory-Type (тип памяти) 2 [радиочастотная метка с единственной памятью (single memory tag)]

Если установлено значение TRUE (ИСТИНА), устройство опроса должно заблокировать идентификатор AFI
Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)

Это идентификатор объекта, связанный с идентификатором AFI в регистре конструкций данных по ИСО/МЭК 15961:2004 *
Object (объект): BYTE STRING (СТРОКА БАЙТОВ)

Это мономорфный идентификатор UII, представленный приложением
Compact-Parameter (параметр уплотнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Единственным допустимым значением является 5 [мономорфный идентификатор UII (Monomorphic-UII)], который должен заставить процессор данных вызвать явным образом определенную схему уплотнения, связанную с идентификатором AFI в регистре конструкций данных по ИСО/МЭК 15961:2004*
Object-Lock (блокировка объекта): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Применяется только для Memory-Type (тип памяти) 2 (радиочастотная метка с единственной памятью)

Если значение — TRUE (ИСТИНА), устройство опроса должно заблокировать соответствующий набор данных
Memory-Bank-Lock (блокировка банка памяти): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Применяется только для Memory-Type (тип памяти) 0 (Банк памяти MB01 радиочастотной метки типа C)

Если значение — TRUE (ИСТИНА), весь банк памяти должен быть заблокирован
Lock-UII-Segment-Arguments (аргументы блокировки сегмента идентификатора UII): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Применяется только для Memory-Type (тип памяти) 1 (сегмент идентификатора UII типа D)

Возможные значения:

Значение	Определение
4	UII Data-Set (набор данных идентификатора UII), исключая слово управления протоколом (PC word) и идентификатор DSFID

* *См.* [*8*].

5	PC word + UII Data-Set (слово управления протоколом+набор данных идентификатора UII) [эта комбинация может применяться к мономорфному идентификатору UII (Monomorphic-UII)]
7	Полный сегмент идентификатора UII, включая код CRC-16

10.24.2 Ответ на команду Write-Monomorphic-UII (записать мономорфный идентификатор UII)

Ответ на команду Write-Monomorphic-UII (записать мономорфный идентификатор UII) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Write-Monomorphic-UII response (ответ на команду «записать мономорфный идентификатор UII»)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 30
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
7	Object-Locked-Could-Not-Modify (заблокированный объект не может быть изменен)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
22	Object-Modified-But-Not-Locked (Объект изменен, но не заблокирован)
25	Password-Mismatch (несоответствие пароля)
26	AFI-Mismatch (несоответствие идентификатора AFI)
33	Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)
34	AFI-Not-For-Monomorphic-UII (идентификатор AFI не для мономорфного идентификатора UII)
35	Monomorphic-UII-OID-Mismatch (несоответствие идентификатора объекта мономорфному идентификатору UII)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.25 Сконфигурировать расширенный идентификатор DSFID (Configure-Extended-DSFID)

10.25.1 Общие положения

Команда Configure-DSFID (сконфигурировать идентификатор DSFID) (см. 10.3) представляет собой структуру для поддержки кодирования метода доступа (Access-Method) и формата данных (Data-Format). Со времени публикации ИСО/МЭК 15961:2004* были добавлены дополнительные функции для создания расширенного идентификатора DSFID, например для сигнализации длины кодированных данных или обнаружения ошибок в данных, которые хранятся в течение длительного периода времени. Команда Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID) используется для установки различных индикаторов, которые кодируются в расширенном идентификаторе DSFID. Поскольку идентификатор DSFID указывает, какое правило кодирования следует за дополнительными функциями, объявленными этой командой, просто определяют дополнительные процессы, которые должны использоваться при кодировании данных.

10.25.2 Команда Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID)

Команда Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID) используется для указания процессору данных (data processor) на кодирование всех соответствующих характеристик идентификатора DSFID в радиочастотную метку. Команду необходимо вызвать прежде, чем будет выполняться какое-либо кодирование в радиочастотной метке.

* *См.* [*8*].

Список конструкций идентификатора DSFID (DSFID-Constructs-list) используется для указания метода доступа (Access-Method) и формата данных (Data-Format). Список конструкций расширенного идентификатора DSFID (Ext-DSFID Constructs-list) используется для установки индикаторов расширенного идентификатора DSFID и указания процессору данных (data processor) на выполнение определенных процедур, например для применения к данным кода CRC.

Аргумент DSFID-Lock (блокировка идентификатора DSFID), если он установлен, применяется ко всему идентификатору DSFID и расширенному идентификатору DSFID.

Команда Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID) имеет следующие аргументы:

- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2);
- DSFID-Lock (блокировка идентификатора DSFID) (см. 7.4.15);
- Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) (см. 11.4);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Аргумент DSFID-Lock (блокировка идентификатора DSFID) должен применяться только после того, как пользователь решит, что все характеристики расширенного идентификатора DSFID (Extended DSFID) могут быть закодированы окончательно.

Команда Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 31
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
DSFID-Constructs-list (список конструкций идентификатора DSFID): List of <DSFID-Constructs> (список <конструкции идентификатора DSFID>)
Ext-DSFID Constructs-list (список конструкций расширенного идентификатора DSFID): [Optional] List of <Ext-DSFID-Constructs> ([ОПЦИОНАЛЬНЫЙ] список <конструкции расширенного идентификатора DSFID>)
DSFID-Lock (блокировка идентификатора DSFID): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)
Если значение аргумента — TRUE (ИСТИНА), устройство опроса должно заблокировать идентификатор DSFID

10.25.3 Ответ на команду Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID)

Ответ на команду Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Configure-Extended-DSFID (сконфигурировать расширенный идентификатор DSFID)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 31
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
4	DSFID-Not-Configured (идентификатор DSFID не сконфигурирован)
5	DSFID-Not-Configured-Locked (блокировка несконфигурированного идентификатора DSFID)
6	DSFID-Configured-Lock-Failed (не удалось заблокировать сконфигурированный идентификатор DSFID)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
37	Data-CRC-Not-Applied (код CRC к данным не применяется)
38	Length-Not-Encoded-In-DSFID (длина не закодирована в идентификаторе DSFID)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.26 Сконфигурировать заголовок блочных записей (Configure-Multiple-Records-Header)

10.26.1 Общие положения

Команда Configure-Multiple-Records-Header (сконфигурировать заголовок блочных записей) должна быть первой вызываемой командой для того, чтобы активировать радиочастотную метку для поддержки блочных записей. Команда может применяться к любой радиочастотной метке с достаточной емкостью памяти, но применяется только к банку памяти 11 сегментированной памяти радиочастотной метки, такой как радиочастотная метка типа С.

10.26.2 Команда Configure-Multiple-Records-Header (сконфигурировать заголовок блочных записей)

Команда Configure-Multiple-Records-Header (сконфигурировать заголовок блочных записей) используется для указания процессору данных (data processor) на кодирование всех соответствующих характеристик заголовка блочных записей (MR-header) в радиочастотной метке. Команда должна быть вызвана до того, как будет происходить кодирование записи в радиочастотной метке.

Аргумент Access-Password (пароль доступа) в команде используется для сопоставления с паролем радиочастотной метки, чтобы разрешить запись данных в радиочастотную метку. Это относится только к радиочастотной метке типа С и может применяться только к этой радиочастотной метке, если такой пароль был ранее закодирован.

Список конструкций идентификатора DSFID (DSFID-Constructs-list) используется для указания метода доступа (Access-Method) (= 4) и формата данных (Data-Format). Если известно, что все записи имеют один и тот же формат данных, определенный как однородное кодирование блочных записей, то значение этого формата данных должно быть закодировано. Если ожидается, что все записи имеют разные форматы данных (Data-Formats), определенные как неоднородное кодирование блочных записей, то должен использоваться формат данных {00001}.

Примечание — Для Multiple-Records Access-Method (метод доступа к блочным записям) интерпретация формата данных {00001} заключается в том, что записи не имеют общего формата данных.

Поскольку требуется метод доступа (Access-Method) {4}, необходимо использовать аргумент Ext-DSFID-Constructs-list (список конструкций расширенного идентификатора DSFID) (см. 11.4), но со следующими особыми вариантами:

- аргумент Memory-Length-Encoding (кодирование объема памяти) необязательно и применяется к каталогу. Возможные значения для битовой строки: 00, чтобы указать, что никакого кодирования не требуется, и 10, чтобы указать, что текущая длина каталога должна вычисляться процессором данных (data processor) и кодироваться в заголовке блочной записи. Если заголовок блочной записи заблокирован, это не может быть использовано;
- аргумент Data-CRC-Indicator (индикатор CRC данных) является необязательным и применяется ко всему компоненту каталога, но рекомендуется, если каталог присутствует. Единственное возможное значение для битовой строки равно 10;
- аргумент Simple-Sensor-Indicator (индикатор простого датчика) является условным и требуется только в том случае, если в радиочастотной метке отсутствуют другие механизмы для объявления наличия простых датчиков;
- аргумент Battery-Assist-Indicator (индикатор наличия батареи) является условным и требуется только в том случае, если в радиочастотной метке отсутствуют другие механизмы для объявления наличия батареи;
- аргумент Full-Function-Sensor-Indicator (индикатор полнофункционального датчика) является условным и требуется только в том случае, если другие механизмы недоступны в радиочастотной метке, чтобы заявить о наличии полнофункциональных датчиков;
- аргумент DSFID-Pad-Bytes (байты-заполнители идентификатора DSFID) не должны использоваться из-за возможности добавления байтов элемента для полного заголовка блочной записи.

Аргумент Directory-Length-EBV8-Indicator (индикатор EBV-8 длины каталога) (см. 7.4.14) является обязательным и используется для указания процессору данных (data processor) на предоставление достаточного пространства в заголовке блочной записи, чтобы разрешить кодирование размера каталога в пределах размеров, установленных приложением.

Аргумент Multiple-Records-Feature-Indicator (индикатор характеристики блочной записи) является обязательным и представляет собой матрицу битов, чтобы предоставить дополнительную информацию о записях, закодированных в логической памяти (более подробную информацию см. в 7.4.39).

Аргумент Sector-Identifier (идентификатор сектора) (см. 7.4.57), если он включен в этот аргумент с ненулевым значением, представляет собой истинный идентификатор сектора, а нулевое значение указывает, что истинный идентификатор сектора закодирован в каждой отдельной записи.

Аргумент Pointer-To-Multiple-Records-Directory (указатель на каталог блочных записей) (см. 7.4.51) является адресом в EBV-8, который показывает начало каталога. Конкретные значения извлекаются из ответов радиоинтерфейса, которые предоставляют подробную информацию об отображении аппаратной памяти радиочастотной метки.

Аргумент Lock-Multiple-Records-Header (блокировка заголовка блочных записей) (см. 7.4.29) используется для определения того, заблокирован ли весь заголовок блочных записей (MR-header) или нет. Если он не заблокирован, возможны следующие варианты:

- оставить разблокированным только поле data length of the directory (длина данных каталога);
- оставить разблокированным только поле number-of-records (число записей);
- оставить разблокированными как поле data length of the directory (длина данных каталога), так и поле number-of-records (число записей).

В незаблокированных случаях процессор данных (data processor) должен обеспечить добавление необходимых заполняющих байтов после предшествующих полей для достижения выравнивания блоков до их блокировки. Он также должен блокировать выравнивание поля(ей) разблокировки.

Команда Configure-Multiple-Records-Header (сконфигурировать заголовок блочных записей) имеет следующие аргументы:

- Access-Password (пароль доступа) (см. 7.4.1);
- Directory-Length-EBV8-Indicator (индикатор EBV-8 длины каталога) (см. 7.4.14);
- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2);
- Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) (см. 11.4);
- Lock-Multiple-Records-Heade (блокировка заголовка блочных записей) (см. 7.4.29);
- Multiple-Records-Features-Indicator (индикатор характеристик блочных записей) (см. 7.4.39);
- Pointer-To-Multiple-Records-Directory (указатель на каталог блочных записей) (см. 7.4.51);
- Sector-Identifier (идентификатор сектора) (см. 7.4.57);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Configure-Multiple-Records-Header (сконфигурировать заголовок блочных записей)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 126 32
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Access-Passwor (пароль доступа): [OPTIONAL] BYTE STRING ([ОПЦИОНАЛЬНЫЙ] СТРОКА БАЙТОВ) (4 байта)
DSFID-Constructs-list (список конструкций идентификатора DSFID): List of <DSFID-Constructs> (список <конструкции идентификатора DSFID>)
Ext-DSFID-Constructs-list (список конструкций расширенного идентификатора DSFID): List of <Ext-DSFID-Constructs> (список <конструкции расширенного идентификатора DSFID>)
Directory-Length-EBV8-Indicator (индикатор EBV-8 длины каталога): [OPTIONAL] INTEGER ([ОПЦИОНАЛЬНЫЙ] ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
1	Единичный байт в формате EBV8
2	Два байта в формате EBV8, даже если длина меньше 128 блоков

Multiple-Records-Features-Indicator (индикатор характеристик блочных записей): BYTE (БАЙТ)
Это карта битов, установленная в данной команде, которая определяет правила для процессора данных (data processor), которым он должен следовать при кодировании отдельных записей
Sector-Identifier (идентификатор сектора): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	Указывает, что идентификатор сектора отличается для разных записей и что истинное значение доступно только для отдельной записи
≠0	Указывает истинное значение сектора, которое применяется ко всем записям

Pointer-To-Multiple-Records-Directory (указатель на каталог блочных записей): EBV-8 VALUE (ЗНАЧЕНИЕ EBV-8)

Это значение основано на ответах устройства опроса на запрос на отображение памяти адресуемой радиочастотной метки

Lock-Multiple-Records-Heade (блокировка заголовка блочных записей): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	Not locked (не блокирован)
1	Completely locked (полностью блокирован)
2	Все компоненты заблокированы, за исключением поля Number-of-records (число записей)

10.26.3 Ответ на команду Configure-Multiple-Records-Header (сконфигурировать заголовок блочных записей)

Ответ на команду Configure-Multiple-Records-Header (сконфигурировать заголовок блочных записей) имеет следующие наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Configure-Multiple-Records-Header (сконфигурировать заголовок блочных записей)

Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = 1 0 15961 127 32

Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
4	DSFID-Not-Configured (идентификатор DSFID не сконфигурирован)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
25	Password-Mismatch (несоответствие пароля)
37	Data-CRC-Not-Applied (код CRC к данным не применяется)
38	Length-Not-Encoded-In-DSFID (длина не закодирована в идентификаторе DSFID)
39	Multiple-Records-Header-Not-Configured (заголовок блочных записей не сконфигурирован)
40	Multiple-Records-Header-Not-Locked (заголовок блочных записей не заблокирован)
41	File-Support-Indicators-Not-Configured (индикаторы поддержки файлов не сконфигурированы)
42	File-Support-Indicators-Not-Locked (индикаторы поддержки файлов не заблокированы)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения: как указано в 9.4.

10.27 Считать блочные записи (Read-Multiple-Records)

10.27.1 Команда Read-Multiple-Records (считать блочные записи)

Команда Read-Multiple-Records (считать блочные записи) дает указание устройству опроса на считывание различных логически структурированных компонентов из радиочастотной метки, сконфигурированной для кодирования блочных записей. Она включает в себя считывание набора из одного или нескольких идентификаторов объекта (Object-Identifiers) и связанных объектов (Objects) из отдельной записи. Для каждой команды должна быть запрограммирована только одна радиочастотная метка, чтобы гарантировать надежность процесса считывания.

Команда применяется к банку памяти 11 радиочастотной метки типа C и к сегменту данных, относящемуся к предмету, в радиочастотной метке типа D.

Аргумент Read-Record-Type (тип считываемой записи) (см. 7.4.52) предоставляет набор вариантов для считывания данных из радиочастотной метки.

Команда Read-Multiple-Records (считать блочные записи) имеет следующие аргументы:

- Max-App-Length (максимальная длина для приложения) (см. 7.4.32);
- Read-Objects (считывание объектов) (см. 11.13);
- Read-Record-Type (тип считываемой записи) (см. 7.4.52);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Read-Multiple-Records (считать блочные записи)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 33
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Read-Record-Type (тип считываемой записи): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	Read-Multiple-Records-Header (считывание заголовка блочных записей)
1	Read-Multiple-Records-Header-Plus-1st-Preamble (считывание заголовка блочных записей и 1-й преамбулы)
2	Read-Multiple-Records-Directory (считывание каталога блочных записей)
3	Read-Preamble-Specific-Multiple-Record (считывание конкретной преамбулы блочной записи)
4	Read-All-Record-OIDs-Specific-Record-Type (считывание всех записей идентификаторов объектов конкретного типа записи)
5	Read-Preamble-Specific-Multiple-Record (считывание конкретной блочной записи идентификаторов объектов)
6	Read-All-Objects-Specific-Multiple-Record (считывание конкретной блочной записи всех объектов)
7	Read-Multiple-Objects-Specific-Multiple-Record (считывание конкретной блочной записи нескольких объектов)
8	Read-1st-Objects-Specific-Multiple-Record (считывание конкретной блочной записи первых объектов)
9	Read-Data-Element-List-Specific-Multiple-Record (считывание конкретной блочной записи списка элементов данных)

Read-Objects-List (список считанных объектов): список <Считанные объекты>

Это относится только к типам:

Read-Preamble-Specific-Multiple-Record (считывание конкретной преамбулы блочной записи) (3),
Read-All-Record-OIDs-Specific-Record-Type (считывание всех записей идентификаторов объектов конкретного типа записи) (4),
Read-All-Objects-Specific-Multiple-Record (считывание конкретной блочной записи всех объектов) (6),
Read-Multiple-Objects-Specific-Multiple-Record (считывание конкретной блочной записи нескольких объектов) (7),
Read-1st-Objects-Specific-Multiple-Record (считывание конкретной блочной записи первых объектов) (8) и
Read-Data-Element-List-Specific-Multiple-Record (считывание конкретной блочной записи списка элементов данных) (9)

См. 7.4.52 для получения подробной информации о том, какие объекты относятся к каждому аргументу Read-Record-Type (тип считываемой записи)
Max-App-Length (максимальная длина для приложения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (1..65535)

Применяется только к типу считывания (Read-Type) (8) и выражается в байтах

10.27.2 Ответ на команду Read-Multiple-Records (считать блочные записи)

Ответ на команду Read-Multiple-Records (считать блочные записи) имеет следующие аргументы и наименования полей:

- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4);
- Multiple-Records-Directory-Structure-List (список структуры каталога блочных записей) (см. 11.9);
- Multiple-Records-Header-Structure-List (список структуры заголовка блочных записей) (см. 11.10);
- Multiple-Records-Preamble-Structure (структура преамбулы блочных записей) (см. 11.11);
- Read-Objects-Response-List (список ответов на считывание объектов) (см. 11.14);
- Read-OIDs-Response-List (список ответов на считывание идентификаторов объектов) (см. 11.15).

Read-Multiple-Records response (ответ на команду «считать блочные записи»)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 33
Multiple-Records-Header-Structure-List (список структуры заголовка блочных записей): [Conditional] List of <Multiple-Records-Header-Structure> ([условная конструкция] список <структура заголовка блочной записи >)
Multiple-Records-Preamble-Structure (структура преамбулы блочных записей): [Conditional] List of <Multiple-records> ([условная конструкция] список <блочные записи>)
Multiple-Records-Directory-Structure-List (список структуры каталога блочных записей): [Conditional] List of <Multiple-Records-Directory-Structure> ([условная конструкция] список <структура каталога блочных записей>)
Read-OIDs-Response-List (список ответов на считывание идентификаторов объектов): [Conditional] List of <Read-OIDs-Response> ([условная конструкция] список <ответов на считывание идентификатора объектов>)
Read-Objects-Response-List (список ответов на считывание объектов): [Conditional] List of <Read-Objects-Response> ([условная конструкция] список <Ответы на считывание объектов (Read-Objects-Response)>)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
48	Multiple-Records-Directory-Not-Present (каталог блочных записей не представлен)
255	Execution-Error (ошибка выполнения)

Дополнительные значения Completion-Code (код завершения) применяются к:
Read-Objects-Response-List (список ответов считанных объектов)
Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

10.28 Удалить блочную запись (Delete-Multiple-Record)

10.28.1 Команда Delete-Multiple-Record (удалить блочную запись)

Команда Delete-Multiple-Record (удалить блочную запись) дает указание устройству опроса либо пометить блочную запись как удаленную, либо физически удалить запись.

Аргумент Delete-MR-Method (метод удаления блочной записи) в команде указывает устройству опроса, какой из двух методов используется.

Если метод предусматривает помечение записи как удаление, то байты, составляющие запись, фактически не удаляются, но запись объекта и его позиция в каталоге (если есть) имеют значения кода, указывающие, что запись больше не обрабатывается как действующая. Если преамбула записи или каталог заблокированы, то команда не может быть вызвана. Если запись является частью иерархической структуры, то сначала необходимо удалить все записи потомков. Если какая-либо из этих преамбул записи или связанная позиция каталога заблокированы, ни одна из записей не может быть удалена.

Если метод предусматривает физическое удаление записи, то в записи меняется все кодирование для того, чтобы сделать эту часть памяти доступной для будущего кодирования. Этот метод не допускается использовать, если какая-либо часть записи заблокирована, но, если преамбула записи не заблокирована, может быть предпринят альтернативный метод.

Аргумент Access-Password (пароль доступа) в этой команде используется для его сопоставления с паролем в радиочастотной метке, чтобы разрешить запись данных в радиочастотную метку, а удаление подразумевает транзакцию записи. Это относится только к радиочастотной метке типа С и может применяться только к этой радиочастотной метке, если ранее такой пароль был закодирован.

Команда Delete-Multiple-Record (удалить блочную запись) имеет следующие аргументы:
- Access-Password (пароль доступа) (см. 7.4.1);
- Object-Identifier (идентификатор объекта) (см. 7.3.3);
- Singulation-Id (идентификатор Singulation-Id) (см. 7.2.1).

Команда Delete-Multiple-Record (удалить блочную запись)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 126 34
Singulation-Id (идентификатор Singulation-Id): BYTE STRING (СТРОКА БАЙТОВ) (0..255)
Access-Password (пароль доступа): [OPTIONAL] BYTE STRING ([ОПЦИОНАЛЬНЫЙ] СТРОКА БАЙТОВ) (4 байта)
Delete-MR-Method (метод удаления блочной записи): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	Пометить запись как удаленную
1	Физически удалить запись

Идентификатор объекта (Object-Identifier): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)
Идентификатор объекта до уровня, где конечная дуга —- это идентификатор экземпляра или иерархического идентификатора

10.28.2 Ответ на команду Delete-Multiple-Record (удалить блочную запись)

Ответ на команду Delete-Multiple-Record (удалить блочную запись) имеет следующие наименования полей:
- Completion-Code (код завершения) (см. 9.3);
- Execution-Code (код выполнения) (см. 9.4).

Ответ на команду Delete-Multiple-Record (удалить блочную запись)
Module-OID (модуль идентификатора объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) = = 1 0 15961 127 34
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
8	Singulation-Id-Not-Found (идентификатор Singulation-Id не обнаружен)
13	Object-Identifier-Not-Found (идентификатор объекта не обнаружен)
25	Password-Mismatch (несоответствие пароля)
49	Record-Not-Deleted-Preamble-Locked (запись не удалена, преамбула заблокирована)
50	Record-Not-Deleted-Directory-Locked (запись не удалена, каталог заблокирован)
51	Record-Not-Deleted-Lower-Level-Preamble-Locked (запись не удалена, преамбула нижнего уровеня заблокирована)
52	Record-Not-Deleted-Encoding-Locked (запись не удалена, кодирование заблокировано)
255	Execution-Error (ошибка выполнения)

Execution-Code (код выполнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения: как указано в 9.4.

11 Аргументы

11.1 Аргумент Add-Objects (добавление объектов)

Аргумент предоставляет список идентификаторов объектов (Object-Identifiers) и объекты (Objects), которые должны быть либо записаны в «чистую» радиочастотную метку, либо добавлены к уже закодированным данным.

Аргумент Add-Objects (добавление объектов) имеет следующие значения аргумента:
- Avoid-Duplicate (предотвращение дублирования) (см. 7.4.6);
- Compact-Parameter (параметр уплотнения) (см. 7.3.6);

- Object (объект) (см. 7.3.5);
- Object-Identifier (идентификатор объекта) (см. 7.3.3);
- Object-Lock (блокировка объекта) (см. 7.3.7).

Аргумент Add-Objects (добавление объектов)
Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)
Avoid-Duplicate (предотвращение дублирования): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)
Если значение — TRUE (ИСТИНА), то устройство опроса должно проверить, что предмет идентификатора объекта уже не кодируется на радиочастотной метке
Object (объект): BYTE STRING (СТРОКА БАЙТОВ)
Compact-Parameter (параметр уплотнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (0..15)
Возможные значения:

Значение	Определение (см. 7.3.6 для более подробной информации)
0	Application-Defined (определено приложением)
1	Compact (уплотнение)
2	UTF8-Data (данные в формате UTF-8)
3	Packed-Objects (упакованные объекты)
4	Tag-Data-Profile (профиль данных радиочастотных меток)

Object-Lock (блокировка объекта): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ))
Если значение — TRUE (ИСТИНА), устройство опроса должно заблокировать соответствующий набор данных

Если Compact-Parameter (параметр уплотнения) имеет значение Packed-Objects (упакованные объекты), то оно должно быть одинаковым для каждого компонента в списке. Значение Object-Lock (блокировка объекта) также должно быть согласованным во всем списке.

11.2 Аргумент DSFID-Constructs (конструкции идентификатора DSFID)

Аргумент DSFID-Constructs (конструкции идентификатора DSFID)
Access-Method (метод доступа): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	No-Directory (без каталога)
1	Directory (каталог)
2	Packed-Objects (упакованные объекты)
3	Tag-Data-Profile (профиль данных радиочастотных меток)
4	Multiple-Records (блочные записи)
5—15	Зарезервировано для расширения при добавлении расширений к идентификатору DSFID

Data-Format (формат данных): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	Not-Formatted (не отформатировано)
1	Full-Featured (полнофункциональный)
2	Root-OID-Encoded (корневой идентификатор объекта закодирован)
3—28	Как опубликовано органом регистрации по ИСО/МЭК 15961-2:2019
29	Для закрытых систем согласно требованиям ИСО/МЭК 15962
30	Для закрытых систем со специальным кодированием
31	Не применяется в качестве назначенного идентификатора DSFID
32—287	Зарезервировано как расширение для формата данных с несколькими байтами (multiple byte Data-Format)

11.3 Аргумент EPC-UII-Memory (память EPC-UII)

Аргумент EPC-UII-Memory (память EPC-UII) Protocol-Control-Word (контрольное слово протокола): BYTE STRING (СТРОКА БАЙТОВ) (2) EPC-Code (кодовое значение EPC): BYTE STRING (СТРОКА БАЙТОВ)

11.4 Аргумент Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID)

Этот аргумент содержит список дополнительных аргументов, которые используются для указания дополнительных функций радиочастотной метки, или процессов, которые процессор данных (data processor) должен использовать для завершения кодирования в радиочастотной метке. Аргумент также используется для некоторых ответов.

Если один или оба бита аргумента Memory-Length-Encoding (кодирование объема памяти) = 1, процессор данных (data processor) должен вычислить соответствующее число блоков и закодировать его в радиочастотной метке в соответствии с синтаксом расширенного идентификатора DSFID.

Если один или оба бита аргумента Data-CRC-Indicator (индикатор CRC данных) = 1, процессор данных (data processor) должен вычислить соответствующий код CRC данных и закодировать его в соответствующих адресах памяти.

Если приложение указывает, что аргумент Simple-Sensor-Indicator (индикатор простого датчика), и/или аргумент Full-Function-Sensor-Indicator (индикатор полнофункционального датчика), и/или аргумент Simple-Sensor-Indicator (индикатор простого датчика) должны быть установлены в значение 1 TRUE (ИСТИНА), процессор данных (data processor) должен закодировать эти настройки независимо от того, действительно ли выбранная функция поддерживается радиочастотной меткой.

Дополнительные байты могут быть закодированы в расширенном идентификаторе DSFID (Extended-DSFID) для будущего кодирования в радиочастотной метке, указав число для аргумента DSFID-Pad-Bytes (байты-заполнители идентификатора DSFID).

Аргументы Length-Of-Encoded-Data (длина закодированных данных) и Memory-Capacity (емкость памяти) возвращаются в качестве аргументов, когда аргумент Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) включен в ответ.

Аргумент Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) имеет следующие аргументы:

- Simple-Sensor-Indicator (индикатор простого датчика) (см. 7.4.7);
- Data-CRC-Indicator (индикатор CRC данных) (см. 7.4.11);
- DSFID-Pad-Bytes (байты-заполнители идентификатора DSFID) (см. 7.4.16);
- Full-Function-Sensor-Indicator (индикатор полнофункционального датчика) (см. 7.4.20);
- Length-Of-Encoded-Data (длина закодированных данных) (см. 7.5.5);
- Memory-Capacity (емкость памяти) (см. 7.5.8);
- Memory-Length-Encoding (кодирование объема памяти) (см. 7.4.35);
- Simple-Sensor-Indicator (индикатор простого датчика) (см. 7.4.58).

Аргумент Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID)
Memory-Length-Encoding (кодирование объема памяти): BYTE STRING (СТРОКА БАЙТОВ)

Возможные значения:

Значение	Определение
00_2	Нет кодированной длины или малый объем памяти (то есть не более 256 бит)
01_2	Объем памяти определен
10_2	Определяется длина кодированных данных
11_2	Определены как объем памяти, так и длина кодирования

Data-CRC-Indicator (индикатор CRC данных): BIT STRING (ДВОИЧНАЯ СТРОКА)

Возможные значения:

Значение	Определение
00_2	Нет данных с циклическим кодом CRC
01_2	Код CRC применяется к каждому отдельному набору данных
10_2	Код CRC применяется только ко всему объему закодированных данных
11_2	Код CRC применяется к каждому набору данных и ко всему объему закодированных данных

Simple-Sensor-Indicator (индикатор простого датчика): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если на радиочастотной метке установлен простой датчик, значение аргумента — TRUE (ИСТИНА)

Simple-Sensor-Indicator (индикатор простого датчика): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если радиочастотная метка снабжена батареей, значение аргумента — TRUE (ИСТИНА)

Full-Function-Sensor-Indicator (индикатор полнофункционального датчика): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если в радиочастотной метке установлен полнофункциональный датчик, значение аргумента — TRUE (ИСТИНА)

DSFID-Pad-Bytes (байты-заполнители идентификатора DSFID): [OPTIONAL (ОПЦИОНАЛЬНЫЙ)] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Это число дополнительных байтов, запрошенных приложением для поддержки дополнительных аргументов, которые будут добавлены позднее, например, Memory-Capacity (емкость памяти) и/или Length-Of-Encoded-Data (длина закодированных данных). Ответ может также включать число заполняющих байтов, чтобы позволить заблокировать расширенный идентификатор DSFID на границе блокирующего блока

Memory-Capacity (емкость памяти): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Этот аргумент включается только в ответ, содержащий аргумент Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID)

Length-Of-Encoded-Data (длина закодированных данных): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Этот аргумент включается только в ответ, содержащий аргумент Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID)

11.5 Аргумент ISO-UIImemory (память ISO-UII)

Аргумент ISO-UIImemory (память ISO-UII)

Protocol-Control-Word (контрольное слово протокола): BYTE STRING (СТРОКА БАЙТОВ) (2)

DSFID (идентификатор DSFID): BYTE (БАЙТ) [OPTIONAL (ОПЦИОНАЛЬНЫЙ)]

Не возвращается, когда идентификатор AFI объявляет, что объект является мономорфным идентификатором UII (Monomorphic-UII)

Read-Objects-Response-List (список ответов считанных объектов): List of <Read-Objects-Response> (список <ответы считанных объектов>)

11.6 Аргумент Item-Related-Add-Objects (добавление объектов, связанных с предметами)

Аргумент команды Item-Related-Add-Objects (добавление объектов, связанных с предметами) имеет ту же функцию и структуру, что и аргумент команды Add-Objects (добавление объектов) (см. 11.1), за исключением того, что он закодирован в сегменте радиочастотной метки, связанном с предметом, которая может адресовать несколько сегментов в транзакциях радиоинтерфейса. Этот специальный аргумент позволяет отличать его от аргумента UII-Add-Objects (добавление объектов в формате идентификатора UII) (см. 11.16).

11.7 Аргумент Item-Related-DSFID-Constructs (конструкции идентификатора DSFID, связанного с предметом)

Аргумент команды (Item-Related-DSFID-Constructs (конструкции идентификатора DSFID, связанного с предметом) данных имеет ту же функцию и структуру, что и аргумент команды DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2), за исключением того, что он закодирован в сегменте радиочастотной метки, связанном с предметом, которая может адресовать несколько сегментов в транзакциях радиоинтерфейса. Этот специальный аргумент позволяет отличать его от аргумента UII-DSFID-Constructs (конструкции UII-DSFID) (см. 11.17).

11.8 Аргумент Multiple-Records-Constructs (конструкции блочных записей)

Этот аргумент предоставляет перечень дополнительных аргументов, которые используются при записи блочных записей в логическую память.

Аргумент логической команды Append-To-Existing-Multiple-Record (добавление к существующей блочной записи) (см. 7.4.4) при значении TRUE (ИСТИНА) используется для объявления, должен ли быть добавлен список объектов данных в ранее существующую запись, с учетом всех аргументов, совпадающих с полями, уже закодированными для блочных записей. Если значение — FALSE (ЛОЖЬ), эту команду следует использовать для создания новой записи.

Аргумент команды Application-Defined-Record-Capacity (емкость записи, определенная приложением) (см. 7.4.5) просто предоставляет приложению выбор: установить объем памяти в заданный размер или позволить процессору данных (data processor) блокировать выравнивание записи. Рекомендуется разрешить процессору данных управлять этим, если только нет причины, по которой приложение устанавливает больший объем памяти, например, чтобы позднее дать возможность закодировать дополнительные предметы. Поэтому аргумент Record-Memory-Capacity (емкость записываемой памяти) требуется только в том случае, если приложение должно контролировать этот аспект памяти.

Аргумент Record-Type-Classification (классификация типа записи) (см. 7.4.56) должен быть выровнен с правилами, определенными в заголовке блочной записи (MR-header). Если используется ненадлежащее значение, команда должна быть отклонена процессором данных.

При построении команды для кодирования записи, которая является частью иерархической структуры, следует ссылаться на стандарт приложения, чтобы гарантировать соответствие иерархических отношений. Аргумент Identifier-Of-My-Parent (родительский идентификатор) (см. 7.4.22) должен основываться на записи, которая уже была предметом предыдущей команды записи. Если есть несоответствие, запись не кодируется.

Аргумент Number-In-Data-Element-List (число экземпляров в списке элементов данных) (см. 7.4.41) применяется только в том случае, если запись является списком элементов данных и используется для предоставления дополнительной информации процессору данных, чтобы гарантировать, что кодируется надлежащее число элементов данных с тем же относительным идентификатором объекта (Relative-OID). Это может быть особенно полезно, когда формат и длина данных изменяются для экземпляров элемента данных.

Аргументы Lock-Record-Preamble (блокировка преамбулы записи) (см. 7.4.30), Update-Multiple-Records-Directory (обновление каталога блочных записей) (см. 7.4.62) и Lock-Directory-Entry (блокировка позиции каталога) (см. 7.4.28) предоставляют инструкции процессору данных и, как правило, связаны с предметной записью. Однако если для аргумента Update-Multiple-Records-Directory (обновление каталога блочных записей) установлено значение TRUE (ИСТИНА), а каталог еще не существует, то следует создать полный каталог. Независимо от значения аргумента, если каталог уже закодирован, то процессор данных должен автоматически полностью обновить каталог.

Аргумент Multiple-Records-Constructs (конструкции блочных записей)

Append-To-Existing-Multiple-Record (добавление к существующей блочной записи): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), процессор данных (data processor) должен проверить соответствие всех аргументов существующей блочной записи и то, что ни одно из значений относительного идентификатора объекта (Relative-OID) уже не существует в текущей версии записи. В случае, когда существующая запись определяется как список элементов данных, ни одно из значений номера элемента списка в этой команде уже не будет закодировано в текущей версии записи. Если установлено значение FALSE (ЛОЖЬ), процессор данных (data processor) должен закодировать запись как новую запись

Application-Defined-Record-Capacity (емкость записи, определенная приложением): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), приложение должно определить аргумент Record-Memory-Capacity (объем памяти для записи). Если FALSE (ЛОЖЬ), процессор данных (data processor) должен просто заблокировать выравнивание записи и закодировать объем записи, необходимый, как минимум, для поддержки записи

Record-Memory-Capacity (емкость памяти для записи): [OPTIONAL (ОПЦИОНАЛЬНЫЙ)] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Это общее число блоков, которые требуются приложению для записи, с любым числом неиспользуемых блоков, которые зарезервированы для дополнительного кодирования в этой записи в будущем

Record-Type-Classification (классификация типа записей): BIT STRING (ДВОИЧНАЯ СТРОКА)

Возможные значения:

Значение	Определение
000_2	автономная запись с дугой экземпляра = 0
001_2	автономная запись с дугой экземпляра > 0

010_2 иерархическая запись, верхний уровень

011_2 иерархическая запись, имеет как родителя, так и потомка(ов)

100_2 иерархическая запись, список элементов данных

101_2 другая иерархическая запись, без потомков

110_2 не относится к этой команде (поскольку она связана с удаленными записями)

111_2 зарезервировано

Identifier-Of-My-Parent (родительский идентификатор): [условная конструкция] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Значение совпадает с иерархическим идентификатором родительской записи

Number-In-Data-Element-List (число экземпляров в списке элементов данных): [условная конструкция] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (1…255)

Значение предоставляется, когда классификация записи (record classification) {100} помогает процессору данных в кодировании соответствующего аргумента Add-Objects (добавить объекты)

Lock-Record-Preamble (блокировка преамбулы записи): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), то преамбула должна быть выровнена по блоку и заблокирована

Update-Multiple-Records-Directory (обновление каталога блочных записей): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), то в каталоге обновляется эта запись, а любые другие записи каталога не обновляются

Lock-Directory-Entry (блокировка позиции каталога): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), позиция каталога для записи должна быть выровнена по блоку и заблокирована

11.9 Аргумент Multiple-Records-Directory-Structure (структура каталога блочных записей)

Этот аргумент ответа предоставляет перечень содержимого каталога блочных записей в таком виде, что информация может быть использована приложением для создания дополнительных команд.

Аргумент Multiple-Records-Directory-Structure (структура каталога блочных записей) содержит следующие аргументы и имена полей:

- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2);
- Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) (см. 11.4);
- Hierarchical-Identifier-Arc (дуга иерархического идентификатора) (см. 7.4.21);
- Identifier-Of-My-Parent (родительский идентификатор) (см. 7.4.22);
- Instance-Of-Arc (дуга экземпляра) (см. 7.4.25);
- Record-Type-Arc (дуга с типом записи) (см. 7.4.55);
- Record-Type-Classification (классификация типа записи) (см. 7.4 56);
- Sector-Identifier (идентификатор сектора) (см. 7.4.57);
- Start-Address-Of-Record (начальный адрес записи) (см. 7.4.59).

Аргумент Multiple-Records-Directory-Structure (структура каталога блочных записей)

DSFID-Constructs-list (список конструкций идентификатора DSFID): [условная конструкция] список <конструкции DSFID (DSFID-Constructs)>

Ext-DSFID-Constructs-list (список конструкций расширенного идентификатора DSFID): [условная конструкция] список <конструкции расширенного DSFID (Ext-DSFID-Constructs)>

Sector-Identifier (идентификатор сектора)): [условная конструкция] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Если идентификатор сектора не был предоставлен заголовком блочной записи, то значение в преамбуле записи действительно

Record-Type-Classification (классификация типа записи): BIT STRING (ДВОИЧНАЯ СТРОКА)

Возможные значения:

Значение	Определение
000_2	автономная запись с дугой экземпляра (instance-of arc)= 0
001_2	автономная запись с дугой экземпляра (instance-of arc)> 0

010_2	иерархическая запись, верхний уровень (top level)
011_2	иерархическая запись, имеет как родителя, так и потомка(ов)
100_2	иерархическая запись, список элементов данных
101_2	другая иерархическая запись, без потомков
110_2	не относится к этой команде (поскольку она связана с удаленными записями)
111_2	зарезервировано

Record-Type-Arc (дуга типа записи): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Instance-Of-Arc (дуга экземпляра): [условная конструкция] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Hierarchical-Identifier-Arc (дуга иерархического идентификатора): [условная конструкция] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Identifier-Of-My-Parent (родительский идентификатор): [условная конструкция] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Значение совпадает с иерархическим идентификатором родительской записи
Start-Address-Of-Record (начальный адрес записи): EBV-8

11.10 Аргумент Multiple-Records-Header-Structure (структура заголовка блочных записей)

Этот аргумент ответа предоставляет список содержимого заголовка блочных записей (MR-header) таким образом, что информация может быть использована приложением для создания дополнительных команд.

Аргумент Multiple-Records-Header-Structure (структура заголовка блочных записей) включает в себя следующие аргументы и наименования полей:

- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2);
- Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) (см. 11.4);
- Multiple-Records-Directory-Length (длина каталога блочных записей) (см. 7.4.38);
- Multiple-Records-Features-Indicator (индикатор характеристик блочных записей) (см. 7.4.39);
- Number-Of-Records (число записей) (см. 7.4.42);
- Pointer-To-Multiple-Records-Directory (указатель на каталог блочных записей) (см. 7.4.51);
- Sector-Identifier (идентификатор сектора) (см. 7.4.57).

Аргумент Multiple-Records-Header-Structure (структура заголовка блочных записей)
DSFID-Constructs-list (список конструкций идентификатора DSFID): List of <DSFID-Constructs> (список <конструкции идентификатора DSFID>)
Ext-DSFID-Constructs-list (список конструкций расширенного идентификатора DSFID): List of <Ext-DSFID-Constructs> (список <конструкции расширенного идентификатора DSFID>)
Multiple-Records-Directory-Length (длина каталога блочных записей): [OPTIONAL (ОПЦИОНАЛЬНЫЙ)] EBV-8
Multiple-Records-Features-Indicator (индикатор характеристик блочных записей): BYTE (БАЙТ)
Представляет собой карту битов, установленную в этой команде, определяющей правила для процессора данных (data processor), которым необходимо следовать при кодировании отдельных записей.
Sector-Identifier (идентификатор сектора): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Возможные значения:

Значение	Определение
0	Указывает, что идентификатор сектора изменяется между записями и что истинное значение доступно только для отдельной записи
≠0	Указывает истинное значение сектора, которое применяется ко всем записям

Pointer-To-Multiple-Records-Directory (указатель на каталог блочных записей): EBV-8
Это значение основано на ответах устройства опроса на отображение памяти адресуемой радиочастотной метки
Number-Of-Records (число записей): EBV-8
Если бит 2 аргумента Multiple-Records-Features-Indicator (индикатор характеристик блочных записей) = 0, то значение этого поля, вероятно, недействительно и будет игнорироваться

11.11 Аргумент Multiple-Records-Preamble-Structure (структура преамбулы блочных записей)

Этот аргумент ответа предоставляет список содержимого преамбулы записи таким образом, что информация может быть использована приложением для создания дополнительных команд.

Аргумент Multiple-Records-Preamble-Structure (структура преамбулы блочных записей) содержит следующие аргументы и имена полей:

- Data-Length-Of-Record (длина записи данных) (см. 7.4.12);
- DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2);
- Encoded-Memory-Capacity (емкость кодированной памяти) (см. 7.4.18);
- Ext-DSFID-Constructs (конструкции расширенного идентификатора DSFID) (см. 11.4);
- Hierarchical-Identifier-Arc (дуга иерархического идентификатора) (см. 7.4.21);
- Identifier-Of-My-Parent (родительский идентификатор) (см. 7.4.22);
- Instance-Of-Arc (дуга экземпляра) (см. 7.4 25);
- Record-Type-Arc (дуга с типом записи) (см. 7.4.55);
- Record-Type-Classification (классификация типа записей) (см. 7.4.56);
- Sector-Identifier (идентификатор сектора) (см. 7.4.57).

Аргумент Multiple-Records-Preamble-Structure (структура преамбулы блочных записей)
DSFID-Constructs-list (список конструкций идентификатора DSFID): List of <DSFID-Constructs> (список <конструкции идентификатора DSFID>
Ext-DSFID-Constructs-list (список конструкции расширенного идентификатора DSFID): [Conditional] List of <Ext-DSFID-Constructs> ([условная конструкция] список <конструкции расширенного идентификатора DSFID>)
Encoded-Memory-Capacity (емкость кодированной памяти): EBV-8

Это размер памяти, зарезервированной для записи, с точки зрения блоков записи

Data-Length-Of-Record (длина записи данных): [условная конструкция] EBV-8

Это размер закодированной записи с точки зрения блоков записи.

Sector-Identifier (идентификатор сектора): [условная конструкция] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Если идентификатор сектора не был предоставлен заголовком блочной записи, то значение в преамбуле записи действительно

Record-Type-Classification (классификация типа записи): BIT STRING (ДВОИЧНАЯ СТРОКА)

Возможные значения:

Значение	Определение
000_2	автономная запись с дугой экземпляра = 0
001_2	автономная запись с дугой экземпляра > 0
010_2	иерархическая запись, верхний уровень
011_2	иерархическая запись, имеет как родителя, так и потомка(ов)
100_2	иерархическая запись, список элементов данных
101_2	другая иерархическая запись, без потомков
110_2	не относится к этой команде (поскольку она связана с удаленными записями)
111_2	зарезервировано

Record-Type-Arc (дуга типа записи): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Instance-Of-Arc (дуга экземпляра): [условная конструкция] INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Hierarchical-Identifier-Arc (дуга иерархического идентификатора): [Conditional] INTEGER ([условная конструкция] ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Identifier-Of-My-Parent (родительский идентификатор): [Conditional] INTEGER ([условная конструкция] ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Это значение совпадает с иерархическим идентификатором родительской записи.

11.12 Аргумент Packed-Objects-Constructs (конструкции упакованных объектов)

Следующие аргументы применимы только к упакованным объектам (Packed-Objects) и игнорируются, если аргумент Access-Method (метод доступа) не является методом Packed-Objects (упакованные объекты).

Аргумент Packed-Objects-Constructs (конструкции упакованных объектов) имеет следующие аргументы:

- Block-Align-Packed-Object (выравнивание упакованных объектов по блоку) (см. 7.4.9);

- Editable-Pointer-Size (размер редактируемого указателя) (см. 7.4.17);
- ID-Type (тип идентификатора) (см. 7.4.24);
- Objects-Offsets-Multiplier (множитель смещения объектов) (см. 7.4.44);
- Packed-Object-Directory-Type (тип каталога упакованного объекта) (см. 7.4.45);
- PO-Directory-Size (размер каталога упакованного объекта) (см. 7.4.48);
- PO-ID Table (таблица идентификаторов упакованных объектов) (см. 7.5.11);
- PO-Index-Length (длина индекса упакованного объекта) (см. 7.4.49).

Аргумент Packed-Objects-Constructs (конструкции упакованных объектов)

PO-ID Table (таблица идентификаторов упакованных объектов): СТРОКА ОКТЕТОВ (OCTET STRING)

ID-Type (тип идентификатора): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (0..15)

Возможные значения:

Значение	Определение (см. подробную информацию в 7.4.24)
0	ID List (список идентификаторов)
1	ID Map (карта идентификаторов)
2—15	Зарезервировано для будущего использования

Packed-Object-Directory-Type (тип каталога упакованного объекта): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (0..15)

Возможные значения:

Значения	Определения (см. подробную информацию в 7.4.45)
0	Этот упакованный объект не является каталогом и не требует каталога
1	Упакованный объект присутствует/отсутствует
2	Поле индекса упакованного объекта
3	Смещение упакованного объекта
4	Размещение указателя упакованного объекта находится в ожидании будущей команды для установки типа каталога
5—15	Зарезервировано для будущего использования

PO-Index-Length (длина индекса упакованного объекта): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) (1…7)

Если этот параметр присутствует, а аргумент Packed-Object-Directory-Type (тип каталога упакованного объекта) равен 2 [Packed-Object index field (поле индекса упакованного объекта)], то реализация должна использовать этот параметр в параметре POIndex Length (длина индекса упакованного объекта), созданном для POIndex Field (поле индекса упакованного объекта) для этого упакованного объекта в каталоге индексов упакованного объекта (PO index directory). Если значение Directory-Type (тип каталога) не равно 2, этот параметр должен быть проигнорирован

Object-Offsets-Multiplier (множитель смещения объекта): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Если этот параметр присутствует, а значение Packed-Object-Directory-Type (тип каталога упакованного объекта) равно 3 [Packed-Object offset (смещение упакованного объекта)] и PO-Index-Length (длина индекса упакованного объекта) присутствует, то реализация должна использовать этот параметр для резервирования числа битов памяти для смещений объекта в секции AuxMap каталога упакованного объекта. Если значение Directory-Type (тип каталога) не равно 3 или параметр PO-Index-Length (длина индекса упакованного объекта) отсутствует, этот аргумент следует игнорировать. Если реализация не может разместить число входных битов для секции AuxMap в каталоге упакованных объектов, реализация должна вернуть Completion-Code (код завершения) Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)

PO-Directory-Size (размер каталога упакованного объекта): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Если этот параметр присутствует, а значение Packed-Object-Directory-Type (тип каталога упакованного объекта) равно 4 (размещение указателя упакованного объекта находится в ожидании будущей команды для установки типа каталога), тогда реализация должна использовать этот аргумент для определения размера созданного указателя каталога для этого упакованного объекта. Если значение Directory-Type (тип каталога) не равно 4, этот параметр должен быть проигнорирован. Если реализация не может назначить размер ввода числа битов для дополнительного упакованного объекта, реализация должна вернуть Completion-Code (код завершения) Insufficient-Tag-Memory (недостаточно памяти радиочастотной метки)

Block-Align-Packed-Objects (выравнивание упакованных объектов по блоку): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), устройство опроса должно гарантировать, что этот упакованный объект начинается с границы блока и что после предыдущего упакованного объекта добавляются все необходимые заполняющие байты (если они есть)

Editable-Pointer-Size (размер редактируемого указателя): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Если задано ненулевое значение, устройство опроса должно пометить упакованный объект как редактируемый и создать указатель на дополнительный упакованный объект с размером аргумента в битах. Если установлено значение, равное нулю, это означает, что дополнительная подсекция не должна включаться

11.13 Аргумент Read-Objects (считывание объектов)

Аргумент Read-Objects (считывание объектов)

Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)

Check-Duplicate (проверка дубликата): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если установлено значение TRUE (ИСТИНА), устройство опроса должно проверить, что есть только одно появление идентификатора объекта, закодированного в радиочастотной метке

11.14 Ответ на считывание объектов (Read-Objects-Response)

Аргумент «ответ на считывание объектов» (Read-Objects-Response)

Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)

Object (объект): BYTE STRING (СТРОКА БАЙТОВ)

Compact-Parameter (параметр уплотнения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Возможные значения:

Значение	Определение
0	Application-Defined (определено приложением)
2	UTF8-Data (данные в формате UTF-8)
14	De-Compacted-Monomorphic-UII (разуплотненный мономорфный идентификатор UII)
15	De-Compacted-Data (разуплотненные данные)

Lock-Status (статус блокировки): BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Если TRUE (ИСТИНА), то набор данных (Data-Set) или упакованный объект (Packed-Object), содержащий идентификатор объекта и объект, заблокирован

Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Эти определения являются дополнительными к ответам, которые включают данный аргумент

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
10	Duplicate-Object (дубликат объекта)
13	Object-Identifier-Not-Found (идентификатор объекта не обнаружен)
15	Object-Not-Read (объект не считан)
35	Monomorphic-UII-OID-Mismatch (несоответствие идентификатора объекта мономорфному идентификатору UII)

11.15 Аргумент Read-OIDs-Response (ответ на считывание идентификаторов объектов)

Аргумент Read-OIDs-Response (ответ на считывание идентификаторов объектов)

Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)

11.16 Аргумент UII-Add-Objects (добавление объектов в формате идентификатора UII)

Аргумент команды UII-Add-Objects (добавление объектов в формате идентификатора UII) имеет ту же функцию и структуру, что и аргумент команды Add-Objects (добавление объектов) (см. 11.1), за исключением того, что он закодирован в сегменте идентификатора UII радиочастотной метки, которая может адресовать

несколько сегментов в тех же транзакциях радиоинтерфейса. Аргумент должен содержать только один Object-Identifier (идентификатор объекта) и Object (объект). Этот специальный аргумент позволяет отличать его от аргумента Item-Related-Add-Objects (добавление объектов, связанных с предметами) (см. 11.6).

11.17 Аргумент UII-DSFID-Constructs (конструкции UII-DSFID)

Аргумент команды UII-DSFID-Constructs (конструкции UII-DSFID) имеет ту же функцию и структуру, что и аргумент команды DSFID-Constructs (конструкции идентификатора DSFID) (см. 11.2), за исключением того, что он закодирован в сегменте идентификатора UII радиочастотной метки, которая может адресовать несколько сегментов в тех же транзакциях радиоинтерфейса. Этот специальный аргумент позволяет отличать его от аргумента Item-Related-DSFID-Constructs (конструкции идентификатора DSFID, связанного с предметом) (см. 11.7).

11.18 Аргумент Write-Responses (ответы на команду записи)

Аргумент Write-Responses (ответы на команду записи)
Object-Identifier (идентификатор объекта): OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)
Completion-Code (код завершения): INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)
Эти определения являются дополнительными к ответам, которые включают этот аргумент

Возможные значения:

Значение	Определение
0	No-Error (нет ошибки)
9	Object-Not-Added (объект не добавлен)
10	Duplicate-Object (дубликат объекта)
11	Object-Added-But-Not-Locked (объект добавлен, но не заблокирован)

Приложение А
(рекомендуемое)

Абстрактный синтаксис и правила кодирования передачи по ИСО/МЭК 15961:2004

А.1 Абстрактный синтаксис

А.1.1 Общие положения

Настоящее приложение включено для обеспечения обратной совместимости абстрактного синтаксиса и правил кодирования с отмененной редакцией этого стандарта 2004 года (ИСО/МЭК 15961:2004*). В приложении Е приведены исходные 16 команд с использованием абстрактного синтаксиса по ИСО/МЭК 15961:2004*. Они могут использоваться как ссылка для сравнения с актуальным видом представления функциональных команд.

Абстрактный синтаксис данного стандарта основан на синтаксисе АСН.1, как определено в ИСО/МЭК 8824-1**. Нотация должна соответствовать указанной в данном стандарте.

А.1.2 Набор знаков

Набор знаков, используемый для определения элемента синтаксиса АСН.1, должен состоять из:

- прописных букв от A до Z;
- строчных букв от a до z;
- цифр от 0 до 9;
- специальных графических знаков: =, { } < . @ ( ) [ ] - ‘ " | & ^ * ; !

Этот набор знаков идентичен тому, который определен в ИСО/МЭК 8824-1**.

А.1.3 Универсальные типы

Система нотаций АСН.1 поддерживает ряд универсальных типов данных, которые являются фундаментальными для синтаксиса, их иногда называют «встроенными типами данных». Чтобы однозначно идентифицировать тип данных, каждому из них в ИСО/МЭК 8824-1** присвоен тег класса. Универсальные типы представлены прописными буквами, например UNIVERSAL. Универсальные типы данных, используемые в ИСО/МЭК 15961:2004*, приведены в таблице А.1 вместе с их тегами класса.

Таблица А.1 — Универсальные типы данных, используемые в ИСО/МЭК 15961:2004*

Универсальный тип данных	Тег класса
BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)	1
INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)	2
OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)	6
OCTET STRING (СТРОКА ОКТЕТОВ)	4
RELATIVE-OID (ОТНОСИТЕЛЬНЫЙ ИДЕНТИФИКАТОР ОБЪЕКТА) (зарезевировано для будущих команд)	13
SEQUENCE & SEQUENCE OF (ПОСЛЕДОВАТЕЛЬНОСТЬ и ПОСЛЕДОВАТЕЛЬНОСТЬ ИЗ)	16

А.1.4 Ссылки на типы данных

В дополнение к универсальным типам данных (Universal Types) синтаксис АСН.1 дает возможность приложениям определять особые типы данных самостоятельно. Когда тип данных определен, ему присваивается соответствующее наименование для указания его на другой тип данных. Ссылки на типы данных начинаются со знака верхнего регистра, и полное имя записывается без пробелов. Существует несколько вариантов представления последовательности знаков. В ИСО/МЭК 15961:2004* используется соглашение о смешанных знаках верхнего и нижнего регистров, например TypeReference (ссылка на тип).

За именем TypeReference (ссылка на тип) следует последовательность из трех знаков «:: =», чтобы отделить имя от его определения.

Ниже представлены примеры имен типов в формате TypeReference (ссылка на тип), которые используются в ИСО/МЭК 15961:2004*:

- applicationFamily (семейство приложений);
- ObjectId (идентификатор объекта);
- StorageFormat (формат хранения);
- TagID (идентификатор радиочастотной метки).

Все наименования типов в формате TypeReference (ссылка на тип), используемые в ИСО/МЭК 15961:2004*, определены в соответствующем пункте настоящего приложения.

---

* *См.* [*8*].

** *См.* [*2*].

А.1.5 Наименования элементов

Для обозначения компонентов, элементов TypeReference (ссылка на тип) или упорядоченного списка используется такая система нотаций, в которой наименование элемента начинается с буквы нижнего регистра, например elementName. Для некоторых элементов требуется последующее добавление знаков либо для TypeReference (ссылка на тип), либо для Universal Type (универсальный тип).

Примерами использования elementNames (имена элементов) в ИСО/МЭК 15961:2004* являются:

- accessMethod (метод доступа);
- applicationFamilyId (идентификатор семейства приложений);
- applicationSubFamily (подсемейство приложений);
- commandCode (кодовое значение команды);
- compactParameter (параметр уплотнения);
- object (объект);
- objectId (идентификатор объекта);
- tagId (идентификатор радиочастотной метки).

Примечание — В ИСО/МЭК 15961:2004* некоторые elementNames (имена элементов) и TypeReference (ссылки на тип) часто представляли собой одно и то же, за исключением того, что первая буква для имени элемента (elementNames) представлялась в нижнем регистре, для ссылки на тип (TypeReference) — в верхнем регистре.

Все elementNames (имена элементов), используемые в ИСО/МЭК 15961:2004*, определены в соответствующих пунктах.

А.1.6 Другие условные обозначения в языке АСН.1

Для демонстрации особенностей используемого синтаксиса АСН.1, ниже приведен достаточно простой пример.

Пример

```
CustomerOrder:: = SEQUENCE {
    orderNumber         INTEGER
    name                    OCTET STRING
    adress                  CustomerAddress
    productDetails SEQUENCE OF SEQUENCE {
                        productCode     OBJECT IDENTIFIER
                        quantity        INTEGER (1..999)
                        },
    urgency                 ENUMERATED {
                        nextDay (0),
                        - - excludes Saturday and Sunday
                        firstClass (1),
                        roadTransport (2),
                        - - typically three days
                        }
}
```

В этом примере используются следующие особенности в системе нотаций:

- двойное двоеточие и знак равенства :: = отделяет имя ссылки на тип TypeReference CustomerOrder от определения;

- в фигурных скобках { }, следующих за SEQUENCE и в конце, указывают, что orderNumber (номер заказа), name (наименование), address (адрес), productDetails (сведения о продукции) и urgency (срочность) — это все элементы ссылки на тип CustomerOrder (заказ покупателя);

- имя элемента address далее указано в ссылке на тип CustomerAddress (для краткости исключено из примера);

- элемент productDetails состоит из двух следующих элементов: productCode (кодовое значение продукции) и quantity (количество). Эта пара элементов повторяется n раз, основываясь на типе данных SEQUENCE OF SEQUENCE. Фигурные скобки { } определяют границы определения типа;

- элемент urgency предлагает одно из трех кодовых значений, 0, 1 или 2, с использованием типа ENUMERATED. Фигурные скобки { } определяют границы определения типа;

- комментариям в этом примере: «excludes Saturday and Sunday» (исключая субботу и воскресенье) и «typically three days» (обычно три дня) — предшествует двойное тире «- -»;

- значение INTEGER для элемента quantity ограничено значениями «(1..999)» и принимает любое значение в диапазоне от 1 до 999, что приводит к ошибке при заказе с количественным значением, равным или большим 1000 предметов с кодовым значением productCode.

А.1.7 Модульная структура синтаксиса АСН.1

В соответствии со стандартами АСН.1 синтаксис, который был использован в ИСО/МЭК 15961:2004*, имеет модульную структуру. Для команд и ответов используются отдельные модули. Каждый модуль содержит:

- уникальное имя;

---

* *См.* [*8*].

- уникальный идентификатор объекта, который ссылается на настоящий стандарт (в соответствии с ИСО/МЭК 8824-1*). Дуга предпоследнего уровня представляет собой либо commandModules (модули команд) (126), либо responseModules (модули ответов (127), чтобы разделять потоки данных. Дуга последнего уровня каждой пары команда/ответ имеет одно и то же имя и значение для установления связи между ними;

- использование ключевых слов DEFINITIONS, BEGIN и END должно соответствовать ИСО/МЭК 8824-1* и позволяет производить обработку модулей средствами компилятора;

- утверждение о том, что настоящий стандарт использует тег «EXPLICIT TAGS», который указывает на то, что все элементы в конечном итоге кодируются как UNIVERSAL TYPES (универсальные типы).

Структура модуля команд имеет следующий общий формат:

Module Name
  {ISO(1) standard(0) rfid-data-protocol (15961) commandModules (126) moduleName(n)}
DEFINITIONS
EXPLICIT TAGS :: =
BEGIN
  CommandName
  - -*assignments*
END

Модуль ответов (responseModule) имеет аналогичный формат.

В каждом модуле все элементы определяются таким образом, чтобы они могли быть сведены до UNIVERSAL TYPES (УНИВЕРСАЛЬНЫЕ ТИПЫ). Это позволяет избежать необходимости реализации любой из функций импорта в рамках модуля.

A.2 Синтаксис передачи данных

A.2.1 Структура кодирования передачи данных

Ниже описывается структура кодирования передачи данных в рамках протокола передачи данных радиочастотной идентификации для управления предметами, первоначально определенная в ИСО/МЭК 15961:2004**.

1 октет(ы) идентификатора типа, который(ые) кодирует(ют) тег типа в стандарте АСН.1 (класс и номер), используемого для определения значения данных;

2 байты длины, которые определяют число байтов, составляющих информационное наполнение (данные);

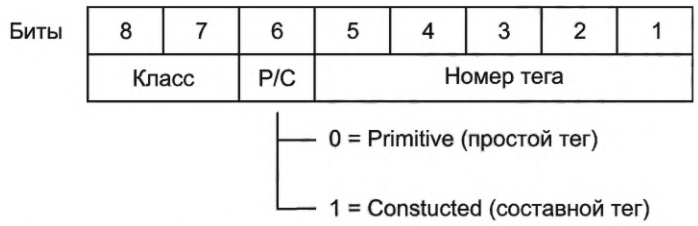
3 байты информационного наполнения (или байты со значениями).

Представленные выше поля иногда называют Type (тип), Length (длина), Value (значение) — формат TLV. Если кодирование основывается на последовательности полей TLV, его называют простым кодированием (Primitive encoding). Значение V может быть триплетом формата TLV, и если используется эта структура кодирования, то его называют составным кодированием (Constructed encoding), например: TL TLV TLV TLV. Выбор между простым и составным кодированием во многом определяется основными правилами кодирования по ИСО/МЭК 8825-1***. Типы конструкторов SEQUENCE и SEQUENCE OF должны использовать составное кодирование (Constructed encoding) для полного соответствия стандартам АСН.1. В противном случае согласно правилам ИСО/МЭК 8825-1* требуется использование простого кодирования или предлагается выбор, в каком случае в стандарте ИСО/МЭК 15961:2004** использовалось только простое кодирование. Данная опция четко определена для каждого из универсальных типов, которые имеют свои правила кодирования BER (Basic Encoding Rules — основные правила кодирования), определенные в последующих статьях.

Перед началом передачи модуль OBJECT IDENTIFIER (идентификатор объекта) должен быть закодирован в формате TLV.

A.2.2 Кодирование идентификатора типа по правилам АСН.1

Идентификатор типа по правилам абстрактной нотации АСН.1 должен быть закодирован для типов, определенных в ИСО/МЭК 15961:2004** в виде одного октета, как показано на рисунке А.1.



где биты 8 и 7 кодируют класс тега в формате АСН.1, бит 6 определяет, является ли тег простым (Primitive) или составным (Constucted), биты с 5 по 1 кодируют номер тега АСН.1, идентифицирующий тип данных, содержащихся в объекте.

Рисунок А.1 — Структура октета идентификатора типа

---

* *См.* [*2*].

** *См.* [*8*].

*** *См.* [*3*].

Двухбитовое значение, описывающее класс, должно быть одним из значений, определенных в таблице А.2. Для типов универсальный (Universal Types), определенный в ИСО/МЭК15961:2004*, его значение должно быть «00_2».

Таблица А.2 — Кодирование класса тега АСН.1

Класс	Бит 8	Бит 7
Universal (универсальный)	0	0
Application (приложение)	0	1
Context-specific (контекстнозависимый)	1	0
Private (частный)	1	1

Единичное значение бита для компонента «П/С» (P/C) (флаг-индикатор «простой/составной») должно быть установлено в «0_2», чтобы указывать на структуры простого кодирования, или должно быть установлено в «1_2» для указания составных структур кодирования.

5-битовое значение для тега АСН.1 (ASN.1) должно кодировать номер тега класса как двоичное целое число со самым старшим битом 5. Теги класса, определенные для ИСО/МЭК 15961:2004*, указаны в таблице А.1.

Пример:
Universal Type = OBJECT IDENTIFIER
ASN.1 Type identifier = 00 0 00110

А.2.3 Кодирование длины

Кодирование длины в ИСО/МЭК 15961:2004* применяется как к простому, так и к составному кодированию. Байты длины должны состоять из одного или нескольких байтов в зависимости от числа числа байтов в информационном наполнении. Если число байтов в информационном наполнении менее или равно 127, то используется один октет длины. Бит 8 должен быть равен «0_2», а биты с 7 по 1 должны кодировать число байтов в информационном наполнении (которое может быть равно нулю) как двоичное целое без знака с битом 7 в качестве самого старшего бита.

Примечание — ИСО/МЭК 15961:2004* ограничил использование кодирования длины по ИСО/МЭК 8825-1**.

Пример
L = 38 кодируется как 00100110_2
Если число байтов в информационном наполнении больше 127, то используются два или более байтов длины. Длина должна быть преобразована в выравненное значение октета, например длина в 201 байт преобразуется в кодовое значение $C9_{16}$ (или 11001001). Это значение кодируется во втором и последующих байтах. Первый октет должен быть закодирован следующим образом:
a. Бит 8 должен быть 1_2.
b. Биты с 7 по 1 должны кодировать число последующих байтов в байтах длины как двоичное целое без знака с битом 7 в качестве самого старшего бита.
c. Значение 11111111_2 не должно использоваться для дальнейшего расширения:
Пример

Длина информационного наполнения = 357

Преобразование в шестнадцатеричное значение = $01\ 65_{16}$

= 00000001_2 01100101_2

Поскольку это 2 байта, первый октет = 100000010_2
Полное кодирование длины:

10000010_2 00000001 01100101_2
= 82 01 65_{16}

А.2.4 Октеты информационного наполнения

Октеты информационного наполнения кодируют значение данных, которое может быть нулевым, одним или несколькими октетами, в зависимости от Universal Type (универсального типа), как указано в последующих подразделах.

---

* *См.* [*8*].
** *См.* [*3*].

А.2.5 Кодирование BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ)

Для кодирования BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ) следует использовать простой тип кодирования в соответствии с ИСО/МЭК 8825-1*. BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ) должно быть закодировано в одном октете. Если BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ) принимает значение FALSE (ЛОЖЬ), октет должен быть равен нулю. Если BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ) принимает значение TRUE (ИСТИНА), то октет должен иметь любое ненулевое значение по выбору отправителя.

А.2.6 Кодирование INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ)

Для кодирования INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) следует использовать простой тип кодирования в соответствии с ИСО/МЭК 8825-1*. INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ) должно быть закодировано в одном или нескольких октетах с использованием следующих процедур.

Для положительных целых чисел и нуля:

a) число целиком преобразуют в двоичное целое число в виде двоичного поля с начальным самым старшим битом;

b) двоичное поле выравнивают по границам октета путем добавления начальных нулевых битов;

c) если бит высокого порядка равен 1_2, добавляют октет 00_{16} в качестве префикса.

Примечание — Бит высшего порядка со значением 0 означает, что кодируется положительное целочисленное значение.

Пример
Целое число 128
Этап 1: 10000000_2
Этап 2: 10000000_2
Этап 3: 00000000_2 10000000_2

Для отрицательных целых чисел кодирование выполняется в соответствии с правилом двойного дополнения:

a) число целиком преобразуют в двоичное целое число в битовом поле с самым старшим битом вначале;

b) битовое поле выравнивают по границам октета путем добавления начальных нулевых битов;

c) двоичное значение из этапа 2 инвертируют (то есть: из 0_2 в 1_2, из 1_2 в 0_2);

d) применяют правило двойного дополнения, добавляющее 1_2 к битовой строке на этапе 2;

e) если бит высшего порядка равен 0, добавляют заполняющий октет FF_{16} в качестве префикса.

Примечание — Бит высшего порядка со значением 1 означает, что кодирование имеет отрицательное целочисленное значение.

Пример
Целое число — 27066
Шаг 1: 1101001_2 10111010_2
Шаг 2: 01101001_2 10111010_2
Шаг 3: 10010110_2 01000101_2
Шаг 4: 10010110_2 01000110_2

В случае декодирования начальный бит кодированного целочисленного значения определяет, является ли значение положительным или отрицательным.

Если это положительное значение, преобразование выполняется над оставшимися битами, причем самый младший бит находится в позиции 0. Целочисленное значение представляет собой сумму значений 2^n, где n — номер позиции или:

$$\sum_{n=0}^{p-1} 2^n.$$

Если это отрицательное значение, преобразование выполняется над оставшимися битами, причем самый младший бит находится в позиции 0. Первый этап заключается в создании десятичного целочисленного значения в виде суммы значений 2^n. На втором этапе оно принимается в качестве положительного десятичного целого, из которого вычитается значение 2^p, где p — номер позиции начального бита, который идентифицирует это как отрицательное целое число. Это представляется формулой:

$$\sum_{n=0}^{p-1} 2^n - 2^p.$$

Пример
10010110_2 01000110_2
1_2 ***указывает отрицательное число***
0010110_2 01000110_2 = ***5702***
2^p 2^{15} ***32768***
5702 – 32768 = –27066

* *См.* [*3*].

А.2.7 Кодирование значения OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА)

Для кодирования значения OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) должен использоваться простой тип кодирования в соответствии с ИСО/МЭК 8825-1*. Значение идентификатора объекта кодируется как серия значений, выравненных по октету, следующим образом:

1. первые две дуги дерева регистрации кодируют как одно целое число по формуле:

$$40f + s,$$

где: f = значение дуги первого уровня;

s = значение дуги второго уровня.

2. значение «n» каждой дуги дополнительного уровня закодировано в битовом поле с выравниванием по октету. Для значений «n» это выполняют следующим образом:

a. при n <128:

десятичное значение преобразуют в двоичное и кодируют в одном октете; таким образом, бит 8 устанавливается равным 0_2;

b. для: 128 ≤ n <16384:

десятичное значение преобразуют в двоичное и подразделяют на две 7-битовые строки: от бита 7 до бита 1, от бита 14 до бита 8. Каждую из этих новых битовых строк кодируют в октет с битом 8 первого октета, установленным в 1_2, и битом 8 последнего октета, установленным в 0_2;

c. для n ≥16384:

десятичное значение преобразуют в двоичное и подразделяют на 7-битовые строки: от бита 7 до бита 1, от бита 14 до бита 8, от бита 21 до бита 15 и т. д. Каждую из этих новых битовых строк кодируют в октете; с битом 8 первого октета, установленным в 1, битом 8 последнего октета, установленным на, 0 и битом 8 промежуточного(ых) октета(ов), установленным(ых) в 1. В приведенном ниже примере показан этот процесс.

Пример

1 Значение $= 91234_{10}$

$= 1\ \ 01100100\ \ 01100010_2$

2 Разделяют на 7-битовые строки

$0000101_2\ \ 1001000_2\ \ 1100010_2$

3 Добавляют биты префикса 0 для последнего(их) октета(ов)

1 для предшествующего(их) октета(ов)

$10000101_2\ \ 11001000_2\ \ 01100010_2$

При использовании этого метода длина каждой дуги компонента для OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) является самоопределяемой. Первый октет всегда определяет первые две дуги. Каждая последующая дуга определяется одним октетом, если начальный бит следующего октета равен 0; и несколькими октетами, если начальный бит равен 1, группа октетов заканчивается октетом со старшим битом, равным 0. Значение дуги кодируется в последовательности 7-битовых значений.

Пример

$[00101000]_2$	$1[1111000]_2$	$0[1001010]_2$	$0[0000001]_2$
(1 x 40) + 0		*15434*	*1*
1 0		*15434*	*1*

Хотя число дуг допускает значение OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА) любой длины, ИСО/МЭК 15961:2004** ограничивает длину закодированного значения не более чем 127 октетами. Это ограничение, предназначенное для удовлетворения требований к кодированию радиочастотной метки и структуре логической памяти.

Ограничение заключается в закодированной длине значения OBJECT IDENTIFIER (ИДЕНТИФИКАТОР ОБЪЕКТА), а не в числе дуг. Следует также понимать, что идентификатор объекта, закодированный в 127 октетах, весьма маловероятен.

А.2.8 Кодирование значения OCTET STRING (СТРОКА ОКТЕТОВ)

Несмотря на то, что основные правила кодирования ИСО/МЭК 8825-1* допускают обе формы кодирования, ИСО/МЭК 15961:2004** поддерживает только простое кодирование значения OCTET STRING (СТРОКА ОКТЕТОВ).

Простое кодирование содержит ноль, один или несколько октетов, равных по значению октетам в значении данных приложения. Закодированные октеты отображаются в том же порядке, в каком они отображаются в значении данных, а самый старший бит октета выравнивается как в кодированных представлениях, так и в представлениях данных.

Для открытых систем стандарты по применению должны гарантировать соответствие последовательности октетов и порядок битов между принимающими и передающими системами.

---

* *См.* [*3*].

** *См.* [*8*].

А.2.9 Кодирование значения SEQUENCE (ПОСЛЕДОВАТЕЛЬНОСТЬ)

Процесс кодирования значений SEQUENCE (ПОСЛЕДОВАТЕЛЬНОСТЬ) должен быть построен в соответствии с ИСО/МЭК 8825-1*. Информационные октеты должны состоять из полного TLV-кодирования одного значения данных каждого из типов, перечисленных в определении типа SEQUENCE (ПОСЛЕДОВАТЕЛЬНОСТЬ) для АСН.1. Значения данных должны быть расположены в порядке их появления в определении. Хотя ИСО/МЭК 8825-1* допускает дополнительные правила для типов с ключевыми словами «OPTIONAL» («ОПЦИОНАЛЬНЫЙ») или «DEFAULT» («ПО УМОЛЧАНИЮ») в определении АСН.1, ИСО/МЭК 15961:2004** требует, чтобы все типы SEQUENCE (ПОСЛЕДОВАТЕЛЬНОСТЬ) появлялись в составном кодировании.

Пример
ASN.1 definition
SEQUENCE {orderNumber OCTET STRING, product OCTET STRING, quantity INTEGER}
со значениями:
{orderNumber "ABC1234", product "widget", quantity "12"}
кодируется как:

T = SEQUENCE	***L***			
30_{16}	14_{16}			
	T = OCTET STRING	***V***	***V***	
	04_{16}	07_{16}	***«ABC1234»***	
	T = OCTET STRING	***V***	***V***	
	04_{16}	06_{16}	***«widget»***	
	T = INTEGER	***V***	***V***	
	02_{16}	01_{16}	$0C_{16}$	

А.2.10 Кодирование значения SEQUENCE OF (ПОСЛЕДОВАТЕЛЬНОСТЬ ИЗ)

Тип SEQUENCE OF (ПОСЛЕДОВАТЕЛЬНОСТЬ ИЗ) имеет тот же тег АСН.1 (UNIVERSAL 16), что и тип SEQUENCE (ПОСЛЕДОВАТЕЛЬНОСТЬ), и поэтому использует те же правила кодирования. Кодирование значения SEQUENCE OF (ПОСЛЕДОВАТЕЛЬНОСТЬ ИЗ) должно быть построено в соответствии с ИСО/МЭК 8825-1*. Октеты информационного наполнения должны состоять из полного кодирования в формате TLV каждого значения, включая кодирование повторяющегося тега класса UNIVERSAL (УНИВЕРСАЛЬНЫЙ) для закодированных элементов.

Пример
ASN.1 definition
sequence of {productCode OCTET STRING}
с тремя значениями:
{productCode «ABC1234», «X6789Y», «PQR12345»}
кодируется как:

T = SEQUENCE	***L***		
30_{16}	$1B_{16}$		
	T = OCTET STRING	***V***	***V***
	04_{16}	07_{16}	***«ABC1234»***
	T = OCTET STRING	***V***	***V***
	04_{16}	06_{16}	***«X6789Y»***
	T = OCTET STRING	***V***	***V***
	02_{16}	08_{16}	***«PQR12345»***

* *См.* [*3*].
** *См.* [*8*].

Приложение В
(рекомендуемое)

Адаптация установленных форматов данных

Настоящий стандарт был основан на том, что его объектно-ориентированный протокол отличается от протоколов, базирующихся на сообщениях и синтаксисе некоторых стандартов по применению автоматической идентификации и сбора данных (АИСД, AIDC). Поэтому базовые объекты данных должны быть представлены способом, соответствующим стандарту по применению, например, с точки зрения:

- объекта данных, поддерживаемого в словаре данных;
- формата данных (например, числовой, буквенно-цифровой), включая его длину;
- комбинаций объектов данных, которые являются допустимыми или запрещенными.

Эти функции выходят за рамки настоящего стандарта и ответственность за них возлагается на приложение.

Необходим некий процесс преобразования, пока прикладные системы не смогут обрабатывать данные и идентификаторы в формате, указанном в настоящем стандарте. Можно иметь два независимых пути реализации: один для записи данных и один для считывания данных.

Для некоторых основных приложений существуют несколько строгих правил для формирования допустимых данных. Для обеспечения соответствия этому формату при использовании существующего синтаксиса на основе сообщений существует программное обеспечение. Пользователи должны убедиться в том, что они реализуют метод записи данных на основе объектов, а сами данные следуют основным правилам. Рисунок В.1 иллюстрирует это схематически.



Рисунок В.1 — Модель потока данных: подготовка данных существующих стандартов приложений

Подобный процесс требуется, когда данные считываются из радиочастотной метки.

Технологии автоматической идентификации и сбора данных, основанные на технологии «однократная запись — многократное считывание» (write-once-read-many-times, WORM), могут полагаться на то, что считываются те данные, которые были записаны. Это означает, что синтаксис сообщения закодирован на носителе данных. Возможности считывания — записи технологии радиочастотной идентификации и объектно-ориентированный характер протокола данных настоящего стандарта означают, что установленный синтаксис должен быть построен на основе структуры идентификатора объекта (Object Identifier). Если необходимо выводить данные с опеределенным синтаксом данных (например, в соответствии с ИСО/МЭК 15434[*]), то требуется модуль преобразования для правильного преобразования набора идентификаторов объектов (Object Identifiers) и объектов данных (data Objects) в формат сообщения, необходимый для применения настоящего стандарта. Это требует инверсии правил преобразования для некоторых данных приложения.

Кроме того, должен быть создан синтаксис сообщения установленного стандарта приложения. Этот процесс обычно требует, чтобы все дуги, кроме конечной дуги идентификатора объекта (Object Identifier), были отброшены, а разделители данных (соответствующие стандарту приложения) были вставлены надлежащим образом. Для получения точных правил следует обратиться к соответствующим стандартам приложений.

Для стандарта приложения логическим шагом разработки является разработка процедур для приема выходных данных на основе синтаксиса передачи.

[*] *См.* [7].

Приложение С
(рекомендуемое)

Связанные объекты данных

С.1 Общие положения

Синтаксис, основанный на сообщениях, может использовать рекурсивные или циклические методы для создания повторяющихся последовательностей связанных данных (например, отдельного значения количества продукции и номеров партий, связанных с разными кодовыми значениями продукции). При анализе полного сообщения синтаксис определяет граничные точки, чтобы атрибуты были надлежащим образом связаны с основным кодовым значением.

При использовании объектно-ориентированной (Object-based) системы [такой, как протокол данных (Data Protocol) настоящего стандарта и ИСО/МЭК 15962], работающей на базовом уровне, существует риск создания ложных ссылок (например, кодовое значение продукции А может быть связано с количеством продукции В). Проблема может быть преодолена с использованием одного из методов, описанных ниже. Методы должны применяться только в том случае, если они включены в стандарт приложения, связанный с элементом, подлежащим управлению. Пояснения ограничивают число построенных элементов данных до 255 на одну радиочастотную метку, но могут быть разработаны различные правила, если требуется большее число конструкций. Любое правило прозрачно для полного протокола данных настоящего стандарта и ИСО/МЭК 15962, поэтому требуется, чтобы обработка была реализована как часть приложения. Для описания надежных способов сохранения объектно-ориентированного процесса сбора данных с использованием древовидной структуры идентификатора объекта в настоящий стандарт включены варианты.

С.2 Метод соединения

Можно создать специальные идентификаторы объектов (Object Identifiers), которые связывают определенный набор атрибутов методом соединения.

Пример
дуга самого низкого уровня 245 =
- порядковый номер 1 октет
- кодовое значение продукции 8 октетов
- количество 1 октет
- номер партии 4 октета

В этом случае первый байт объекта, порядковый номер, отличает одни подобные данные объекта от других.

Используя этот метод, различным расположениям основных элементов для создания конкатенированной конструкции будут присвоены различные конечные узлы. Таким образом, соединение продукция + количество + срок годности будет иметь значение дуги наименьшего уровня, отличное от значения для соединения продукция + количество + партия.

Этот метод более подходит, когда необходимо создать фиксированные комбинации элементов и зафиксировать длину каждого объекта.

С.3 Метод расширения идентификатора объекта

Базовый идентификатор объекта (basic Object Identifier) может быть расширен путем добавления новой конечной дуги, причем со значением «связывание» (linking).

Пример
Следующие три элемента должны быть связаны:
product code (кодовое значение продукции) — конечная дуга 48
quantity (количество) — конечная дуга 17
batch (партия) — конечная дуга 20
Предполагают, что есть два разных вида продукции, данные которых закодированы в радиочастотной метке. Таким образом, применяют расширения ссылок 1 и 2. Шесть отдельных идентификаторов объектов (Object-Identifiers) кодируют следующим образом:
... 48 1
.... 48 2
.... 17 1
.... 17 2
.... 20 1
.... 20 2

Значение расширения (extension) используется для связывания различных объектов как логической комбинации.

Этот метод более подходит, когда необходимо создать множество различных комбинаций элементов и длина по крайней мере одного объекта может отличаться для различных случаев употребления.

Метод расширения для связывания объектов и связанных с ними физических объектов аналогичен схеме В для обеспечения безопасности данных (см. приложение С.2). Поэтому для любого одного идентификатора объекта (Object-Identifier) этот метод применяется только к защите данных или к связыванию физических объектов.

Приложение D
(рекомендуемое)

Вопросы обеспечения безопасности данных

D.1 Общие положения

Несмотря на то, что защита данных выходит за рамки требований настоящего стандарта и ИСО/МЭК 15962, предлагаются следующие рекомендации, чтобы показать, как функции протокола данных могут использоваться для обеспечения защиты данных.

D.2 Проблемы с идентификатором объекта

Зашифрованные данные должны быть связаны с собственным уникальным идентификатором объекта (Object-Identifier). Это гарантирует, что только авторизованные пользователи смогут распознавать зашифрованные данные. Сам объект просто появляется с аргументом уплотнения (Compact-Parameter), равным 0 Application-Defined (определено приложением) (см. 7.3.6).

Один из методов создания идентификатора объекта (Object-Identifier), называемый для последующих ссылок схема А, заключается в том, чтобы иметь конечную дугу на том же уровне, что и все остальные конечные дуги в прикладной системе. Это системный уровень обеспечения безопасности данных, и он требует, чтобы все авторизованные пользователи знали, что данные зашифрованы. Однако правила могут публично не объявляться.

Другой метод для создания уникального идентификатора объекта (Object-Identifier) для идентификации зашифрованных данных, называемый для последующих ссылок схема В, — это расширение идентификатора объекта (Object-Identifier) открытых (незашифрованных) данных путем добавления дополнительной дуги нижнего уровня. Этот метод может быть применен в двух направлениях между отправителем и авторизованным(и) пользователем(ями) или на системных уровнях для всех авторизованных пользователей. Этот метод также допускается использовать для определения типа шифрования, выбранных ключей и т. д.

Пример

0 1 15961 nn nn	***Plain Object (открытый объект)***
0 1 15961 nn nn 1	***Encrypted Object (шифрованный объект)***
0 1 15961 nn nn 2	***Encryption type (тип шифрования)***
0 1 15961 nn nn 3	***Key (ключ)***

Схема В аналогична схеме, предложенной для связывания объектов и связанных с ними физических объектов (см. приложение С.3). Поэтому для любого идентификатора объекта (Object-Identifier) этот метод применяется либо для обеспечения безопасности данных, либо для связывания физических объектов.

D.3 Объект данных

Для объектов (Object), содержащих данные приложения, аргумент Compact-Parameter (параметр уплотнения) принимает значение 0 Application-Defined (определено приложением) после шифрования.

Основной объект (basic Object) должен быть расширен для включения предопределенного поля данных или подписи уполномоченной записывающей стороны. Это поможет обеспечить целостность данных, поскольку любая несанкционированная модификация зашифрованного объекта без знания закрытого ключа приведет к тому, что подпись станет недействительной. Это поможет определить, что объект был изменен без полномочий.

Если для зашифрованных данных назначается полностью другой идентификатор объекта (Object-Identifier) (схема А, приведенная выше), его, возможно, необходимо развернуть, чтобы он также содержал дополнительные незашифрованные октеты, которые определяют схему шифрования и/или выбор из набора ключей.

D.4 Использование идентификатора радиочастотной метки (Tag ID)

В качестве компонента в защищенной системе допускается использовать уникальный Tag ID (идентификатор радиочастотной метки), определенный для различных типов, приведенных в серии стандартов ИСО/МЭК 18000*. Идентификатор Tag ID должен быть уникальным для радиочастотной метки, чтобы отличать ее от всех остальных. Он обычно создается на ранней стадии изготовления радиочастотной метки с использованием более надежных методов, чем те, что могут использоваться для записи данных на карту логической памяти (Logical Memory Map). Таким образом, его можно использовать для повышения достоверности данных. Из-за возможной путаницы с идентификатором Singulation-ID протокола данных (Data Protocol) (который может использовать идентификатор Tag ID) идентификатор Tag ID, встроенный в интегральную схему, упоминается как идентификатор 18000 Tag ID (идентификатор радиочастотной метки по ИСО/МЭК 18000) для остальной части данного приложения.

Если идентификатор 18000 Tag ID также действует в качестве идентификатора Singulation-ID как часть системной информации, идентификатор 18000 Tag ID предоставляется на ранней стадии связи с радиочастотной меткой. Если идентификатор 18000 Tag ID не указан как часть системной информации, для его считывания требуются дополнительные команды.

---

* *См.* [*10*] — [*18*].

Один из методов заключается в объединении значения идентификатора 18000 Tag ID с базовыми данными объекта и шифровании всего расширенного объекта. При расшифровании авторизованным пользователем идентификатор 18000 Tag ID в расширенном объекте (expanded Object) можно сравнить с реальным идентификатором 18000 Tag ID, чтобы убедиться, что они идентичны. Это может быть применено к любой Схеме А или В для создания Object-Identifier (идентификатора объекта), описанного выше.

Другой метод заключается в том, чтобы использовать идентификатор 18000 Tag ID для изменения исходного ключа для шифрования и расшифрования. Этот метод может применяться к двоичным ключам, таким как ключи алгоритма DES, следующим образом: идентификатор 18000 Tag ID покомпонентно складывается по модулю 2 с исходным ключом, и тем самым формируется новый ключ.

Прежде чем использовать этот подход, разработчики должны убедиться, что этот тип модификации ключа не подрывает метод шифрования.

D.5 Рекомендации по использованию алгоритмов шифрования с открытым ключом

Алгоритмы с открытым ключом используют ключи большей длины, чтобы обеспечить тот же уровень обеспечения безопасности, что и симметричные алгоритмы. Например, 64-битовый симметричный ключ и 512-битовый асимметричный ключ обеспечивают примерно одинаковый уровень обеспечения безопасности. Такая длина ключа может препятствовать использованию алгоритмов шифрования с открытым ключом в радиочастотных метках с меньшей памятью.

При использовании алгоритма шифрования с открытым ключом данные будут скомпрометированы, если для их зашифрования используют закрытый ключ, а для расшифрования применяют открытый ключ. Аналогично целостность данных будет нарушена, если для формирования подписи используют открытый ключ, а для проверки подписи — закрытый ключ. Для одновременного обеспечения безопасности и целостности данных необходимо использовать разные ключевые пары или другие криптографические механизмы.

Открытый ключ не должен быть закодирован в радиочастотной метке, если она не заблокирована, поскольку неавторизованная сторона может нарушить целостность данных, перезаписав открытый ключ другим ключом и используя другой соответствующий закрытый ключ для шифрования измененных данных в объекте.

Приложение Е
(рекомендуемое)

Исходные команды и ответы с использованием абстрактного синтаксиса языка АСН.1

Е.1 Общие положения

В следующих подразделах приведены исходные 16 модулей с использованием абстрактного синтаксиса языка АСН.1. Перекрестные ссылки, иногда с изменением имен, приведены в подразделах в нормативной части, которые теперь заменяют те, что были в исходных модулях.

Е.2 Модули ConfigureAfiModules (сконфигурировать идентификатор AFI)

Модули ConfigureAfiModules (сконфигурировать идентификатор AFI) состоят из commandModule (модуля команды) и связанного с ними responseModule (модуля ответа), которые дают указание устройству опроса на запись идентификатора AFI (applicationFamilyId) в радиочастотную метку. Основное требование этой команды состоит в том, что для каждой команды должна быть запрограммирована только одна радиочастотная метка. Это необходимо для обеспечения надежного процесса конфигурирования, особенно в средах, где может присутствовать более одного типа радиочастотной метки.

Абстрактный синтаксис модулей ConfigureAfiModules (сконфигурировать идентификатор AFI) приведен в следующем примере:

```
-- Configure AFI (Сконфигурировать индентификатор AFI)
-- ConfigureAfiCommand (команда «сконфигурировать идентификатор AFI»)
-- дает указание устройству опроса на запись идентификатора AFI
-- (идентификатора семейства приложений, включая подсемейство) в
-- радиочастотную метку. Устройство опроса должно блокировать
-- идентификатор AFI, если Lock flag (флаг блокировки) установлен в
-- значение TRUE (ИСТИНА).
ConfigureAfiCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) configureAfi(1)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
ConfigureAfiCommand : := SEQUENCE {
     tagId                OCTET STRING(SIZE(0..255)),
                         -- См. подробную спецификацию в 7.2.1
                         -- (теперь переименован в идентификатор
                         -- Singulation-Id). Идентификатор TagId должен
                         -- предоставляться драйвером радиочастотных меток
                         -- для однозначной идентификации радиочастотной
                         -- метки, по крайней мере, на период транзакции.
    applicationFamilyId ApplicationFamilyId,
    afiLock               BOOLEAN
                           -- Если TRUE, устройство опроса должно
                           -- заблокировать идентификатор AFI
    }
ApplicationFamilyId : := SEQUENCE {
     applicationFamily INTEGER {
                         all (0), -- адресует все семейства
                         -- Значения 1 - 8 зарезервированы для определения
                         -- подкомитетом SC17
                         afiBlock9(9),
                         afiBlockA(10),
                         afiBlockB(11),
                         afiBlockC(12)
                         -- Значения 9 - 12 определены согласно приложению
                         -- В настоящего стандарта значения 13 - 15
                         -- зарезервированы для определения ИСО/МЭК
                         }  (0..15),
     applicationSubFamily INTEGER {
                         all(0),-- Это значение не должно кодироваться в
                         -- радиочастотную метку, а должно использоваться в
```

```
                        -- команде, чтобы указать, что устройство опроса
                        -- адресует все подсемейства в рамках выбранного
                        -- семейства.
                        -- Примечание - Это малоприменимо для
                        -- настоящего протокола данных, но допустимо для
                        -- совместимости с командами смарт-карт
                        -- подкомитета SC17
                        asf1-annex (1),
                            -- значения 1 - 15, для applicationFamily
                            -- 9 - 12, определенные согласно приложению B
                            -- настоящего стандарта
                        asf2-annex (2),
                        asf3-annex (3),
                        asf4-annex (4),
                        asf5-annex (5),
                        asf6-annex (6),
                        asf7-annex (7),
                        asf8-annex (8),
                        asf9-annex (9),
                        asfA-annex (10),
                        asfB-annex (11),
                        asfC-annex (12),
                        asfD-annex (13),
                        asfE-annex (14),
                        asfF-annex (15)
                        } (0..15)

-- Идентификатор ApplicationFamilyId сохраняется на радиочастотной метке
-- как одиночный OCTET в рамках информационной системы.
-- Идентификатор ApplicationFamilyId дает возможность группировать
-- радиочастотные метки в соответствии с конкретными семействами и дает
-- возможность приложению выборочно адресовать любое такое семейство.
-- Изготовители радиочастотных меток могут реализовать механизмы в
-- драйвере радиочастотных меток и радиоинтерфейсе конкретно для
-- выборочной адрессации радиочастотных меток с помощью идентификатора
-- ApplicationFamilyId.
}
END

ConfigureAfiResponse
     {iso(1) standard(0) rfid-data-protocol(15961)responseModules(127) configureAfi(1)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

ConfigureAfiResponse : := SEQUENCE {
      completionCode INTEGER {
                     noError(0),
                     afiNotConfigured(1),
                     afiNotConfiguredLocked(2),
                     afiConfiguredLockFailed(3),
                     tagIdNotFound(8),
                     executionError(255)
                     },
     executionCode INTEGER
                     -- См. 9.4 и примечания в этом синтаксисе для
                     -- полного списка кодов executionCodes
}
END
```

Следующие имена элементов (elementNames), используемые в этих модулях, определены в другом месте настоящего стандарта как:

- afiLock (блокировка идентификатора AFI) (см. 7.4.3);
- applicationFamily (семейство приложений) (см. 7.2.2);
- applicationFamilyId (идентификатор семейства приложений) (см. 7.2.2);
- applicationSubFamily (подсемейство приложений) (см. 7.2.2);
- completionCod (код завершения) (см. 9.3);
- executionCode (код выполнения) (см. 9.4);
- tagId (идентификатор радиочастотной метки) (см. 7.2.1).

E.3 Модули ConfigureStorageFormatModules (сконфигурировать формат хранения)

Модули ConfigureStorageFormatModule (сконфигурировать формат хранения) состоят из commandModule (модуля команды) и связанного с ним responseModule (модуля ответа), которые дают указание устройству опроса на запись storageFormat (формата хранения) [включает accessMethod (метода доступа) и dataFormat (формат данных)] в радиочастотную метку. Указанная команда также дает указание устройству опроса на инициализацию карты логической памяти (Logical Memory Map) радиочастотной метки, удаляя любые данные, уже сохраненные там. Основное требование этой команды состоит в том, что для каждой команды должна быть запрограммирована только одна радиочастотная метка. Это необходимо для обеспечения надежного процесса конфигурирования, особенно в средах, где может присутствовать более одного типа радиочастотной метки.

Если accessMethod (метод доступа) [включенный в storageFormat (формат хранения)] указан как directory (каталог), устройство опроса должно создать исходную структуру каталога.

Абстрактный синтаксис на языке ACH.1 для модулей ConfigureStorageFormatModules (сконфигурировать формат хранения) приведен в следующем примере:

```
-- Configure StorageFormat (сконфигурировать формат хранения)
-- ConfigureStorageFormatCommand (команда «сконфигурировать формат
-- хранения») дает указание устройству опроса на запись StorageFormat
-- (формат хранения) в радиочастотную метку и инициализацию карты
-- логической памяти (logical memory map) радиочастотной метки.
-- Устройство опроса должно очистить всю память приложения и, если
-- формат directory (каталог) указан с помощью формата хранения
-- (StorageFormat), должно создать начальную структуру directory
-- (каталог).Устройство опроса должно блокировать формат хранения
-- (StorageFormat), если флаг блокировки (Lock flag) установлен в
-- значение TRUE (ИСТИНА).
ConfigureStorageFormatCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) configureStorageFormat(2)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
ConfigureStorageFormatCommand : := SEQUENCE {
      tagId                    OCTET STRING(SIZE(0..255)),
                               -- См. подробную спецификацию в 7.2.1
                               -- (теперь переименован в идентификатор
                               -- Singulation-Id) для подробной
                               -- спецификации. Идентификатор TagId
                               -- должен предоставляться драйвером
                               -- радиочастотной метки Tag Driver
                               -- для однозначной идентификации
                               -- радиочастотной метки, по крайней мере,
                               -- на период транзакции.
      storageFormat           StorageFormat,
      storageFormatLock       BOOLEAN
                              -- Если TRUE, устройство опроса должно
                              -- блокировать формат хранения
                              -- (StorageFormat)
      }
StorageFormat : := SEQUENCE {
        accessMethod INTEGER {
                              noDirectory(0),
                              directory(1),
                              selfMappingTag(2) -- Доступ к объектам
                              -- осуществляется посредством команд высокого
                              -- уровня к радиочастотной метке, а внешняя
```

```
                              -- структура памяти внутри радиочастотной метки
                              -- не определяется
                              }   (0..3),
dataFormat  INTEGER {
            notFormatted(0),  -- Не отформатирована в
                              -- соответствии с настоящим стандартом
            fullFeatured(1),  -- Поддерживает любой тип
                              -- данных, основанный на полном
                              -- идентификаторе объекта (OID)
            rootOidEncoded(2),  -- Поддерживает любой тип
                                -- данных с общим корневым идентификатором
                                -- объекта (root-OID)
            iso15434(3),  -- Корневой идентификатор объекта
                          -- (root-OID) определен как {1 0 15434}
            iso6523(4),   -- Поддерживает данные,
                          -- принадлежащие к одному или более указателям
                          -- международного кода, соответствующих
                          -- ИСО/МЭК 6523-1, корневой идентификатор
                          -- объекта (root-OID) определен как (1 0 6523)
            iso15459(5),  -- Поддерживает уникальные
                          -- идентификаторы объектов, соответствующих
                          -- ИСО/МЭК 15459, корневой идентификатор
                          -- объекта (root-OID) определен как (1 0
                          -- (1 0 15459)
            iso15961Combined(8),  -- Поддерживает комбинации
                                  -- форматов по ИСО/МЭК 15961, корневой
                                  -- идентификатор объекта (root-OID) определен
                                  -- как {1 0 15961}
            ean-ucc(9),   -- Поддерживает данные системы
                          -- EAN-UCC, корневой идентификатор объекта
                          -- (root-OID) определен как {1 0 15961 9}
            di(10),   -- Поддерживает идентификаторы данных
                      -- (как указано в ИСО/МЭК 15418),  корневой
                      -- идентификатор объекта (root-OID)
                      -- представляет собой {1 0 15961 10}
            iata(12)  -- Поддерживает элементы данных
                      -- обработки багажа IATA (International Air
                      -- Transport Association), корневой
                      -- идентификатор объекта (root-OID) определен
                      -- как {1 0 15961 11}
            } (0..31)
}
END
ConfigureStorageFormatResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) configureStorageFormat(2)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
ConfigureStorageFormatResponse : := SEQUENCE {
      completionCode     INTEGER {
                                 noError(0),
                                 storageFormatNotConfigured(4),
                                 storageFormatNotConfiguredLocked(5),
                                 storageFormatConfiguredLockFailed(6),
                                 tagIdNotFound(8),
                                 executionError(255)
                                 },
      executionCode      INTEGER
                                 -- См. 9.4 и примечания к этому синтаксису
                                 -- для полного списка значений кода
                                 -- выполнения executionCodes)
}
END
```

Следующие имена элементов (elementNames), используемые в этих модулях, определены в другом месте настоящего стандарта, как:

- accessMethod (метод доступа) (см. 7.2.4);
- completionCode (код завершения) (см. 9.3);
- dataFormat (формат данных) (см. 7.2.5);
- executionCode (код выполнения) (см. 9.4);
- storageFormat (формат хранения) (см. 7.2.3);
- storageFormatLock (блокировка формата хранения) (см. 7.4.15);
- tagId (идентификатор радиочастотной метки) (см. 7.2.1).

E.4 Модули InventoryTagsModules (инвентаризировать радиочастотные метки)

Модули InventoryTagsModules (инвентаризировать радиочастотные метки) состоят из commandModule (модуль команды) и связанного с ними responseModule (модуль ответа), которые дают указание устройству опроса на идентификацию определенного набора радиочастотных меток, присутствующих в его рабочем поле.

Абстрактный синтаксис на языке ACH.1 для модулей InventoryTagsModules (инвентаризировать радиочастотные метки) приведен в примере ниже.

Команда требует, чтобы значение идентификатора applicationFamilyId (идентификатор семейства приложений) указывалось для выбора радиочастотных меток, принадлежащих к определенному классу, обычно содержащему данные, принадлежащие определенному домену и/или содержащие определенный идентификатор объекта (objectId).

Второй, дополнительный, критерий выбора [identifyMethod (метод идентификации)] определяет, сколько радиочастотных меток, соответствующих указанному критерию выбора идентификатора applicationFamilyId, необходимо определить до того, как будет предоставлен ответ. Механизм, который может быть использован для обнаружения любой радиочастотной метки, входящей в рабочую зону, заключается в том, чтобы установить аргумент inventoryAtLeast (инвентаризация не менее, чем) равным 1. Особые условия могут быть подтверждены только за счет частичной инвентаризации, то есть с использованием аргументов inventoryAtLeast (инвентаризация не менее, чем) или inventoryNoMoreThan (инвентаризация не более, чем). Согласование известного числа предыдущих транзакций (например, для идентификации того, что все предметы, предназначенные для размещения в контейнере, на самом деле находятся там), может быть достигнуто с использованием аргумента inventoryExactly (инвентаризация в точном соответствии). Подробная информация о аргументах приведена в модуле в приведенном ниже примере.

Ответ состоит из аргумента numberOfTagsFound (число обнаруженных радиочастотных меток) и ключевых идентификаторов каждой радиочастотной метки по ее идентификатору tagId.

```
-- InventoryTags (инвентаризировать радиочастотные метки)
-- InventoryTagsCommand (команда «инвентаризировать радиочастотные
-- метки») поручает устройству опроса считать и идентифицировать все
-- радиочастотные метки, присутствующие в рабочей зоне. Каждая
-- радиочастотная метка однозначно идентифицируется своим
-- идентификатором.

InventoryTagsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) inventoryTags(3)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

InventoryTagsCommand : := SEQUENCE {
   applicationFamilyId       ApplicationFamilyId,
   identifyMethod            INTEGER {
                                inventoryAllTags (0),
                                inventoryAtLeast (1),
                                inventoryNoMoreThan (2),
                                inventoryExactly (3)
                                }(0..15),
   numberOfTags              INTEGER (0..65535)
   }

-- Идентификатор ApplicationFamilyId разделяет принципиально разные
-- типы данных приложения (см. 7.2.2) и возможно конкретные
-- идентификаторы объектов. Указание шестнадцатиричного значения xx
-- (где x представляет собой ненулевое значение) выбирает только
-- радиочастотные метки, которые имеют требуемое содержание данных.
-- Указание шестнадцатиричного значения 00 выбирает все радиочастотные
-- метки; это может быть использовано для того, чтобы произвести полную
-- инвентаризацию.
```

```
-- Указание шестнадцатеричного значения 0x или x0 (где x ненулевое
-- значение) не может быть логически правильным, поскольку
-- радиочастотные метки от разных приложений, а значение x имеет
-- различное значение.
-- Если для аргумента identifyMethod (метод идентификации)
-- установлено значение inventoryAllTags (инвентаризация всех
-- радиочастотных меток), устройство опроса должно выполнить считывание
-- всех радиочастотных меток в зоне своего действия. Значение
-- numberOfTags (число радиочастотных меток) несущественно и приложение
-- должно установить его в ноль.
-- Если для аргумента identifyMethod (метод идентификации) установлено
-- значение inventoryAtLeast (инвентаризация не менее, чем), устройство
-- опроса должно выполнить считывание всех радиочастотных меток в зоне
-- своего действия и (возможно) ожидать, пока не определится число
-- радиочастотных меток, равное numberOfTags (число меток). Если для
-- numberOfTags (число меток) установлено значение 1, то устройство
-- опроса будет ожидать обнаружения первой радиочастотной метки. Это
-- механизм для ожидания появления радиочастотной метки на входе в поле
-- считывания. Если для numberOfTags установлено значение более 1, то
--  устройство опроса будет ожидать до тех пор, пока не будет
-- обнаружено указанное число радиочастотных меток.
-- Если для аргумента identifyMethod (метод идентификации) установлено
-- значение inventoryNoMoreThan (инвентаризация не более, чем), то
-- устройство опроса будет инициировать инвентаризацию радиочастотных
-- меток, присутствующих в его рабочем поле, и возврат ответа с числом
-- радиочастотных меток менее или равным numberOfTags (число
-- радиочастотных меток). Устройство опроса может прервать процесс
-- инвентаризации, когда будет достигнуто значение numberOfTags, или
-- может продолжить считывание до тех пор, пока все радиочастотные
-- метки не будут считаны.
--    Примечание - Это может быть ограничено радиоинтерфейсом и
--    антиколлизионным механизмом.
-- Если для аргумента identifyMethod (метод идентификации)установлено
-- значение inventoryExactly (инвентаризация в точном соответствии),
-- устройство опроса должно начать инвентаризацию радиочастотных меток,
-- присутствующих в его зоне действия, и возвращать ответ с числом
-- радиочастотных меток, равным numberOfTags. Этот параметр команды
-- используется для подтверждения фактического числа снабженных
-- радиочастотными метками предметов в контейнере. Устройство опроса
-- будет ждать, пока указанное число радиочастотных меток не будет
-- обнаружено. Устройство опроса может прекратить процесс
-- инвентаризации, когда будет достигнуто число numberOfTags, или
-- может продолжить процесс инвентаризации до тех пор, пока все
-- радиочастотные метки не будут считаны.
--    Примечание - Это может быть ограничено радиоинтерфейсом и
--    антиколлизионным механизмом.
-- Выполнение этой команды со значениями аргументов inventoryAtLeast
-- (инвентаризация не менее, чем) и inventoryExactly (инвентаризация в
-- точном соответствии) может указать устройству опроса на ожидание,
-- пока достаточное число радиочастотных меток не войдет в его рабочее
-- поле; также ответ на команду не может быть инициилизирован после
-- этой задержки. Возможность реализации этого возложена на приложение.
ApplicationFamilyId : := SEQUENCE {
  applicationFamily              INTEGER(0..15),
  applicationSubFamily           INTEGER(0..15)
}
-- Кодовые значения определяются в команде ConfigureAfiCommand
-- (сконфигурировать идентификатор AFI).

TagId : := OCTET STRING(SIZE(0..255))
          -- См. 7.2.1 для подробного описания
END
```

```
InventoryTagsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) inventoryTags(3)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

InventoryTagsResponse : := SEQUENCE {
   completionCode INTEGER {
                        noError(0),
                        failedToReadMinimumNumberOfTags(23),
                        -- например, это может быть из-за тайм-аута
                        -- failedToReadExactNumberOfTags(24),
                        -- например, это может быть из-за тайм-аута executionError(255)
},
executionCode INTEGER,
                        -- см. 9.4 и примечания к этому синтаксису для
                        -- полного списка значений кода выполнения
                        -- (executionCodes)
numberOfTagsFound INTEGER (1..65535),
identities        SEQUENCE OF TagId
}
TagId : := OCTET STRING(SIZE(0..255))
                        -- См. 7.2.1 для подробного описания
END
```

Следующие имена элементов (elementNames), используемые в этих модулях, определены в другом месте настоящего стандарта как:

- applicationFamily (семейство приложений) (см. 7.2.2);
- applicationFamilyId (идентификатор семейства приложений) (см. 7.2.2);
- applicationSubFamily (подсемейство приложений) (см. 7.2.2);
- completionCode (код завершения) (см. 9.3);
- executionCode (код выполнения) (см. 9.4);
- identifyMethod метод (идентификации) (см. 7.4.23);
- identities (ключевые идентификаторы) (см. 7.5.3);
- numberOfTags (число радиочастотных меток) (см. 7.4.43);
- numberOfTagsFound (число обнаруженных радиочастотных меток) (см. 7.5.10);
- tagId (идентификатор радиочастотной метки) (см. 7.2.1).

E.5 Модули AddSingleObjectModules (добавить единичный объект)

Модули AddSingleObjectModules (добавить единичный объект) состоят из модуля команды (commandModule) и связанного с ними модуля ответа (responseModule), которые дают указание устройству опроса на запись объекта (object), идентификатора объекта (objectId) и соответствующих параметров в карту логической памяти (Logical Memory Map) радиочастотной метки. Аргументы команды могут использоваться для блокировки идентификатора объекта (objectId), объекта (object) и связанных с ними параметров, а также для проверки того, что идентификатор объекта (objectId) еще не закодирован в радиочастотной метке. Для каждой команды должна быть запрограммирована только одна радиочастотная метка, чтобы гарантировать надежность процесса записи.

Абстрактный синтаксис на языке ACH.1 для исходного модуля AddSingleObjectModules (добавить единичный объект) приведен в следующем примере:

```
-- Add Single Object (добавить единичный объект)
-- AddSingleObjectCommand (команда добавить единичный объект) дает
-- указание устройству опроса на запись объекта, его идентификатора OID
-- и связанных с ними параметров в карту логической памяти
-- радиочастотной метки.
-- Примечание - Существует также команда AddMultipleObjectsCommand
-- (добавить несколько объектов).
-- Если checkDuplicate flag (флаг проверки дубликата) установлен в
-- значение TRUE (ИСТИНА), устройство опроса должно проверить, прежде
-- чем добавить объект, что объекта с тем же идентификатором (OID)
-- не существует. Если такие объекты существуют, устройство опроса
-- не должно выполнять функцию Add Object (добавить объект)
-- и возвратить соответствующий Completion Code (код завершения).
```

```
-- Если Lock flag (флаг блокировки) установлен в значение TRUE
-- (ИСТИНА), устройство опроса должно заблокировать идентификатор
-- объекта (ObjectId), объект (Object), его схему уплотнения
-- (compaction scheme) и связанные с ними параметры в карте логической
-- памяти радиочастотной метки.
AddSingleObjectCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) addSingleObject(4)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
AddSingleObjectCommand : := SEQUENCE {
tagId                  OCTET STRING (SIZE (0..255)),
                       -- См. 7.2.1 для подробного описания
objectId               OBJECT IDENTIFIER, -- Полное значение OID
avoidDuplicate         BOOLEAN,
                       -- Если установлено TRUE, проверяются
                       -- идентификаторы объектов дубликатов
                       -- (duplicate objectId)
object                 OCTET STRING,
compactParameter       INTEGER {
                         applicationDefined(0),
                       -- Объект не должен обрабатываться по
                       -- правилам уплотнения данных согласно
                       -- ИСО/МЭК 15962 и остается неизменным
                       compact(1),
                       -- Уплотнение объекта эффективно
                       -- настолько,насколько это определено
                       -- правилами уплотнения ИСО/МЭК 15962
                        utf8Data(2)
                       -- Данные были внешне преобразованы из
                       -- 16-битового кодового набора знаков
                       -- в кодированную строку знаков UTF-8.
                       -- Объект не должен обрабатываться по
                       -- правилам уплотнения ИСО/МЭК 15962 и
                       -- остается неизменным
                         }(0..15),
                       BOOLEAN
                       -- Если значение TRUE, то устройство
                       -- опроса должно зафиксировать
                       -- идентификатор объекта (ObjectId),
                       -- объект (Object), его схему уплотнения и
                       -- другие функции в логической карте
                       -- памяти
}
END
AddSingleObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) addSingleObject(4)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

AddSingleObjectResponse : := SEQUENCE {
     completionCode INTEGER {
                            noError (0),
                            tagIdNotFound (8),
                            objectNotAdded (9),
                            duplicateObject (10),
                            objectAddedButNotLocked (11),
                            executionError (255)
                            },
```

```
    executionCode  INTEGER
                           -- См. 9.4 и примечания к этому синтаксису для
                           -- полного списка значений executionCodes (коды
                           -- выполнения)
    }
END
```

Следующие имена элементов (elementNames), используемые в этих модулях, определены в другом месте настоящего стандарта как:

- avoidDuplicate (избежание дублирования) (см. 7.4.6);
- compactParameter (параметр уплотнения) (см. 7.3.6);
- Completion-Code (код завершения) (см. 9.3);
- executionCode (код выполнения) (см. 9.4);
- object (объект) (см. 7.3.5);
- objectId (идентификатор объекта) (см. 7.3.3);
- objectLock (блокировка объекта) (см. 7.3.7);
- tagId (идентификатор радиочастотной метки) (см. 7.2.1).

E.6 Модули DeleteObjectModules (удалить объект)

Модули DeleteObjectModules (удалить объект) состоят из модуля команды (commandModule) и связанного с ними модуля ответа (responseModule), которые дают указание устройству опроса на удаление определенного объекта (object), идентификатора объекта (objectId) и связанных с ним параметров. Только одна радиочастотная метка и только один идентификатор объекта должны быть запрограммированы для каждой команды, чтобы гарантировать надежность процесса удаления. Функция удаления требует удаления идентификатора объекта, связанного с ним объекта и прекурсора (precursor) из карты логической памяти.

Абстрактный синтаксис на языке ACH.1 для модулей DeleteObjectModules (удалить объект) приведен в следующем примере:

```
-- Delete Object (удалить объект)
-- DeleteObjectCommand (команда «удалить объект») дает указание
-- устройству опроса на удаление объекта с указанным идентификатором
-- объекта (OID) из карты логической памяти радиочастотной метки. Это
-- означает, что последующая команда на считывание объекта вернет
-- значение objectNotFound (объект не обнаружен). Эта процедура может
-- не осуществиться, если объект заблокирован. Если это обнаружено как
-- причина, ответ вернет соответствующее значение completionCode.
-- Если checkDuplicate flag (флаг проверки дубликата) установлен в
-- значение TRUE (ИСТИНА), устройство опроса должно проверить до
-- удаления запрошенного объекта, что имеется только один объект с
-- запрошенным идентификатором объекта. Если устройство опроса
-- обнаруживает, что несколько объектов имеют одинаковые идентификаторы
-- объекта, то оно не выполняет функцию DeleteObject (удаление объекта)
-- и возвращает соответствующее значение Completion-Code.
-- Если checkDuplicate flag (флаг проверки дубликата) установлен в
-- значение FALSE (ЛОЖЬ), устройство опроса должно удалить первое
-- вхождение объекта, указанное его идентификатором объекта (OID).
--   Примечание - Это аргумент, который фактически не обеспечивает
--   защиту от дубликата идентификатора объекта. Его следует
--   использовать только тогда, когда существует большая уверенность,
--   что дубликаты отсутствуют.
DeleteObjectCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) deleteObject(5)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
DeleteObjectCommand : := SEQUENCE {
    TagId              OCTET STRING(SIZE(0..255)),
                       -- См. 7.2.1 для подробного описания
    objectId           OBJECT IDENTIFIER, -- полное значение OID
                       -- Это инициирует удаление идентификатора
                       -- объекта (ObjectId) и ассоциированного объекта
```

```
                -- (Object)
    checkDuplicate      BOOLEAN
                      -- Если checkDuplicate (флаг проверки дубликата)
                      -- установлен в значение TRUE (ИСТИНА),
                      -- устройство опроса должно проверить, что есть
                      -- только одно вхождение идентификатора объекта
                      -- (ObjectId)

    }
END
DeleteObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) deleteObject(5)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
DeleteObjectResponse : := SEQUENCE {
    completionCode          INTEGER {
                                noError (0),
                                tagIdNotFound (8),
                                duplicateObject (10),
                                objectNotDeleted (12),
                                objectIdNotFound (13),
                                objectLockedCouldNotDelete (14),
                                executionError (255)
},
    executionCode           INTEGER
                    -- См. 9.4 и примечание к этому синтаксису
                    -- для полного списка значений кода
                    -- выполнения (executionCodes)
    }
END
```

E.7 Модули ModifyObjectModules (измененить объект)

Модули ModifyObjectModules (измененить объект) состоят из модуля команды (commandModule) и связанного с ним модуля ответа (responseModule), которые дают указание устройству опроса на выполнение трех связанных процессов:

a) считывание полной карты логической памяти (Logical Memory Map) из радичастотной метки;

b) удаление указанного идентификатора объекта (objectId), объекта (object) и связанного с ним прекурсора (precursor). Если обнаружены дублированные экземпляры, процесс прерывается;

c) запись идентификатора объекта (objectId), модифицированного объекта и реструктурированного прекурсора.

Объект (object) должен быть расположен в другой области карты логической памяти. Подобная ситуация может возникнуть, если аргумент objectLock (блокировка объекта) в команде установлен в значение TRUE (ИСТИНА). В обоих случаях процессор обработки протокола данных (Data Protocol Processor) по ИСО/МЭК 15962 будет контролировать процесс перемещения (relocation). Только одна радиочастотная метка и только один идентификатор объекта должны быть запрограммированы для каждой команды, чтобы гарантировать надежность процесса изменения.

Абстрактный синтаксис на языке АСН.1 для модулей ModifyObjectModules (изменить объект) приведен в следующем примере:

```
-- Modify Object (изменить объект)
-- ModifyObjectCommand (команда «измененить объект») дает указание
-- устройству опроса на изменение объекта, идентификатора объекта и
-- связанных с ними параметров в карте логической памяти радиочастотной
-- метки. Это выполняется путем удаления указанного объекта и связанных
-- с ним параметров и записи новых значений. Чтобы достичь этого,
-- следует считывать из радиочастотной метки полную карту логической
-- памяти. Наличие любых экземпляров дубликатов идентификаторов
-- объектов приводит к прерыванию процесса.
-- Если объект уже заблокирован, его нельзя изменить, и при этом должно
-- быть возвращено соответствующее значение Completion-Code.
```

```
-- Если строка байтов, представляющая измененные данные при подготовке
-- к кодированию в карте логической памяти, имеет такую же длину, как и
-- предыдущее кодирование, измененное значение обычно записывается на
-- те же адреса.
-- Если строка байтов короче предыдущего кодирования, то смещение
-- (offset) должно быть закодировано.
-- Если строка байтов длиннее предыдущего кодирования, тогда она должна
-- быть размещена в другой области логической карты памяти под
-- управлением процессора обработки протокола данных (Data Protocol
-- Processor) по ИСО/МЭК 15962.
ModifyObjectCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) modifyObject(6)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
ModifyObjectCommand  : := SEQUENCE {
   TagId             OCTET STRING(SIZE(0..255)),
                     -- См. 7.2.1 для подробного описания
   objectId          OBJECT IDENTIFIER, -- Полное значение OID
   object            OCTET STRING,
   compactParameter  INTEGER {
                           applicationDefined(0),
                           -- Объект не должен обрабатываться по
                           -- правилам уплотнения ИСО/МЭК 15962 и
                           -- остается неизменным
                           compact(1),
                           -- Уплотненный объект – по наиболее
                           -- эффективным правилам уплотнения
                           -- ИСО/МЭК 15962
                           utf8Data(2)
                           -- Данные были внешне преобразованы из 16-
                           -- битового кодированного набора знаков в
                           -- строку знаков UTF-8. Объект не должен
                           -- обрабатываться по правилам уплотнения
                           -- ИСО/МЭК 15962 и остается неизменным
                           }(0..15),
objectLock     BOOLEAN
 -- Если установлено TRUE (ИСТИНА), то устройство
 -- опроса должно зафиксировать идентификатор
 -- объекта (ObjectId), объект, его схему уплотнения
 -- и другие параметры в логической карте памяти
}
END
ModifyObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) modifyObject(6)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

ModifyObjectResponse : := SEQUENCE {
                                INTEGER {
                                   noError (0),
                                   objectLockedCouldNotModify (7),
                                   tagIdNotFound (8),
                                   duplicateObject (10),
                                   objectNotModified (21),
                                   objectModifiedButNotLocked (22),
                                   executionError (255)
},
```

```
      ecutionCode                INTEGER
                                 -- См. 9.4 и примечание к этому
                                 -- синтаксису для полного списка
                                 --  значений executionCodes
    }
END
```

Е.8 Модули ReadSingleObjectModules (считать одиночный объект)

Модули ReadSingleObjectModules (считать одиночный объект) включают в себя модуль команды (commandModule) и связанный модуль ответа (responseModule), которые дают указание устройству опроса на считыватние идентификатора объекта (objectId), объекта (object) и связанных параметров из карты логической памяти радиочастотной метки. Аргументы команды могут использоваться для проверки, не является ли идентификатор объекта (objectId) на радиочастотной метке дублированным. Только одна радиочастотная метка может быть запрограммирована для каждой команды, чтобы обеспечить надежный процесс считывания.

Абстрактный синтаксис на языке АСН.1 для модулей ReadSingleObjectModules (считать одиночный объект) приведен в следующем примере:

```
-- Read Single Object (считать одиночный объект)
-- ReadSingleObjectCommand (команда «считать одиночный объект») дает
-- указание устройству опроса на считывание объекта (Object),
-- определенного через идентификатор объекта (Object Identifier), из
-- радиочастотной метки, заданной идентификатором TagId.
--    Примечание - Существует команда считывания нескольких
--    объектов (ReadMultipleObjectsCommand).
-- Если значение checkDuplicate flag (флаг проверки дубликата) FALSE
-- (ЛОЖЬ), устройство опроса должно вернуть первый обнаруженный объект,
-- имеющий запрошенный идентификатор объекта без проверки на наличие
-- дубликатов.
-- Если значение checkDuplicate flag (флаг проверки дубликата) TRUE
-- (ИСТИНА), устройство опроса должно проверить наличие дубликатов
-- объектов, имеющих запрошенный идентификатор объекта. Если
-- присутствует более одного объекта с запрошенным идентификатором
-- объекта, устройство опроса должно вернуть первый обнаруженный
-- объект, имеющий запрошенный идентификатор объекта, и указать на
-- присутствие дубликатов с соответствующим кодом завершения.
ReadSingleObjectCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) readSingleObject(7)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
ReadSingleObjectCommand : := SEQUENCE {
tagId             OCTET STRING(SIZE(0..255)),
                  -- См. 7.2.1 для подробного описания
objectId          OBJECT IDENTIFIER, -- полное значение OID
checkDuplicate    BOOLEAN
                  -- Если установлено TRUE (ИСТИНА), устройство
                  -- опроса должно проверить, что имеется только
                  -- один экземпляр идентификатора (ObjectId)
}
END
ReadSingleObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) readSingleObject(7)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

ReadSingleObjectResponse    ::= SEQUENCE {
completionCode       INTEGER {
                        noError (0),
                        tagIdNotFound (8),
```

```
                        duplicateObject (10),
                        objectIdNotFound (13),
                        objectNotRead (15),
                        executionError (255)
                        },
executionCode           INTEGER ,
                        -- см. 9.4 и примечание к этому синтаксису
                        -- для полного списка значений кода
                        -- выполнения (executionCodes)
object                  OCTET STRING,
compactParameter        INTEGER {
                        applicationDefined(0),
                        -- Объект не был первоначально
                        -- закодирован по правилам уплотнения
                        -- данных ИСО/МЭК 15962 и в таком
                        -- виде передан от приложения-источника,
                        -- может потребовать дополнительной
                        -- обработки принимающим приложением.
                  utf8Data(2),
                        -- Данные были внешне преобразованы из
                        -- 16-битового кодового набора знаков в
                        -- строку знаков UTF-8. Объект должен
                        -- быть обработан с помощью внешнего
                        -- декодера знаков UTF-8.
                  de-compactedData(15)
                        -- Объект был первоначально закодирован
                        -- по правилам уплотнения данных ИСО/МЭК
                        -- 15962 и разуплотнен на этой операции
                        -- считывания и восстановлен в
                        -- своем первоначальном формате.
                  }(0..15),
lockStatus              BOOLEAN
                        -- Если TRUE, объект заблокирован
}
END
```

E.9 Модули ReadObjectIdsModules (считать идентификаторы объектов)

Модули ReadObjectIdsModules (считать идентификаторы объектов) включают в себя модуль команды (commandModule) и связанный модуль ответа (responseModule), которые дают указание устройству опроса на считывание всех идентификаторов объектов (objectIds) из радиочастотных меток. Данный модуль может быть использован как более селективная команда для считывания отдельного объекта или для определения дубликатов идентификаторов объекта, для чего может быть вызвана обслуживающая процедура. Допускается ответ в виде пустого списка идентификаторов объектов, если в карте логической памяти радиочастотной метки нет идентификаторов объектов. Для каждой команды необходимо запрограммировать только одну радиочастотную метку, чтобы гарантировать надежность процесса считывания.

Абстрактный синтаксис на языке АСН.1 модулей ReadObjectIdsModules (считать идентификаторы объектов) представлен в следующем примере:

```
-- Read Object Ids (считать идентификаторы объектов)
-- ReadObjectIdsCommand (команда «считать идентификаторы объектов»)
-- дает указание устройству опроса на считывание всех идентификаторов
-- объектов (Object Ids) из радиочастотной метки с указанным
-- идентификатором TagId. Объекты могут быть считаны индивидуально
-- командой ReadObjectCommand (считать объекты).
-- Если имеются дубликаты идентификаторов объектов, устройство
-- опроса должно вернуть их как блочные экземпляры OID в списке
-- обнаруженных идентификаторов объектов (objectIdsFound).
-- Если радиочастотная метка не имеет идентификаторов объектов, то
-- значение objectIdsFound (обнаруженные идентификаторы объектов)
-- будет возвращено пустым; и команда будет выполнена без ошибки.
ReadObjectIdsCommand
{iso (1) standard (0) rfid-data-protocol (15961) commandModules (126) readObjectIds (8)}
DEFINITIONS
```

```
EXPLICIT TAGS : :=
BEGIN
      TagId: :=   OCTET STRING(SIZE(0..255))
                  -- См. 7.2.1 для подробного описания
END

ReadObjectIdsResponse
{iso (1) standard (0) rfid-data-protoco (15961) responseModules (127) readObjectIds (8)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

ReadObjectIdsResponse : :=  SEQUENCE {
    completionCode           INTEGER {
                                  noError (0),
                                  tagIdNotFound (8),
                                  executionError (255)
                                  },
    executionCode            INTEGER ,
                             -- см. 9.4 и примечания к этому синтаксису
                              -- для полного списка значений кода
                              -- выполнения (executionCodes)
    objectIdsFound            SEQUENCE OF ObjectId
    }

ObjectId : := OBJECT IDENTIFIER -- Полное значение идентификатора OID

END
```

E.10 Модули ReadAllObjectsModules (считать все объекты)

Модули ReadAllObjectsModules (считать все объекты) включают в себя модуль команды (commandModule) и связанный модуль ответа (responseModule), которые дают указание устройству опроса считать целиком карту логической памяти радиочастотной метки и дать ответ на команду считывания в полностью структурированном виде, который идентифицирует каждый экземпляр: идентификатора объекта (objectId), объекта (object), параметра уплотнения (compactParameter), статуса блокировки (lockStatus). Только одна радиочастотная метка может быть запрограммирована для каждой команды, чтобы обеспечить надежность процесса считывания.

Абстрактный синтаксис на языке АСН.1 для модулей ReadAllObjectsModules (считать все объекты) представлен в следующем примере:

```
-- Read All Objects (считать все объекты)
-- ReadAllObjectsCommand (команда «считать все объекты») дает указание
-- устройству опроса на считывание всех объектов, идентификаторов
-- объектов и связанных параметров из указанной радиочастотной метки.
-- Устройство опроса должно вернуть все объекты и их параметры из
-- радиочастотной метки, определенной ее идентификатором TagId. Если
-- присутствуют дубликаты идентификаторов объектов, устройство опроса
-- должно вернуть, как блочные экземпляры идентификатора объекта,
-- объекты и связанные параметры в списке объектов.
ReadAllObjectsCommand
{iso (1) standard (0) rfid-data-protocol (15961) commandModules (126) readAllObjects (9)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
TagId: :=              OCTET STRING(SIZE(0..255))
                       -- См. 7.2.1 для подробного описания
END
ReadAllObjectsResponse
{iso (1) standard (0) rfid-data-protoco l(15961) responseModules (127) readAllObjects
(9)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
```

```
ReadAllObjectsResponse : :=  SEQUENCE {
     completionCode INTEGER {
                         noError (0),
                         tagIdNotFound (8),
                         objectsNotRead (16),
                         executionError (255)
},
executionCode       INTEGER,
                     -- см. 9.4 и примечание к этому синтаксису
                     -- для полного списка значений кода
                     -- выполнения (executionCodes)
objects             SEQUENCE OF SEQUENCE {
    objectId           OBJECT IDENTIFIER,
    object             OCTET STRING,
    compactParameter       INTEGER {
                            applicationDefined(0),
                            -- Объект не был первоначально
                            -- закодирован по правилам уплотнения
                            -- данных ИСО/МЭК 15962 и в таком виде
                            -- передан от приложения-источника,
                            -- может потребовать дополнительной
                            -- обработки принимающим приложением.
                            utf8Data(2),
                            -- Данные были внешне преобразованы из
                            -- 16-битового кодового набора знаков
                            -- в строку знаков UTF-8. Объект
                            -- должен быть обработан с помощью
                            -- внешнего устройства декодирования
                            -- знаков UTF-8.
                            de-compactedData(15)
                            -- объект был первоначально
                            -- закодирован по правилам уплотнения
                            -- данных ИСО/МЭК 15962 и разуплотнен
                            -- при считывании и восстановлен в
                            -- своем первоначальном формате.
                            }(0..15),
      lockStatus    BOOLEAN
                    -- Если TRUE (ИСТИНА), объект заблокирован
      }
    }
END
```

E.11 Модули ReadLogicalMemoryMapModules (считать карту логической памяти)

Модули ReadLogicalMemoryMapModules (считать карту логической памяти) включают в себя модуль команды (commandModule) и связанный модуль ответа (responseModule), которые дают указание устройству опроса на считывание целиком карты логической памяти (Logical Memory Map) радиочастотной метки и выдачу ответа неструктурированным способом (т. е. с возвратом значений закодированных байтов). Обработка данных через процессор обработки протокола данных (Data Protocol Processor) не является частью этой команды считывания, поэтому отдельные идентификаторы объектов (objectIds), объекты (objects), параметр уплотнения (compactParameter) и статус блокировки (lockStatus) не могут быть идентифицированы напрямую. Кроме того, если структура directory (каталог) была определена accessMethod (метод доступа), то структура должна быть включена в ответ, но не должна отличаться от других байтов в карте логической памяти. Основная функция этой команды предназначена для диагностики, но она также может использоваться для других функций, где требуется считывание полного содержимого карты логической памяти. Командой должна быть запрограммирована только одна радиочастотная метка, чтобы гарантировать надежность процесса считывания.

Абстрактный синтаксис на языке АСН.1 для модулей ReadLogicalMemoryMapModules (считать карту логической памяти) приведен в следующем примере:

```
-- Read Logical Memory Map (считать карту логической памяти)
-- ReadLogicalMemoryMap (команда «считать карту логической памяти»)
-- дает указание устройству опроса вернуть все содержимое карты
-- логической памяти радиочастотной метки, определенной ее
```

```
-- идентификатором TagId. Команда отличается от команды ReadAllObjects
-- (считать все объекты) тем, что передает исходную строку байтов
-- приложения без обработки процессором протокола данных (Data Protocol
-- Processor). Это может быть использовано для диагностики.

ReadLogicalMemoryMapCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126)
readLogicalMemoryMap(10)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
     TagId : :=   OCTET STRING(SIZE(0..255))
                  -- См. 7.2.1 для подробного описания
END

ReadLogicalMemoryMapResponse
{iso (1) standard (0) rfid-data-protocol (15961) responseModules (127)
readLogicalMemoryMap (10)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

ReadLogicalMemoryMapResponse : :=  SEQUENCE {
     completionCode    INTEGER {
                              noError (0),
                              tagIdNotFound (8),
                              readIncomplete (19),
                              executionError (255)

                              },
     executionCode     INTEGER,
                       -- см. 9.4 и примечания к этому синтаксису для
                       -- полного списка значений executionCodes
     logicalMemoryMap  OCTET STRING
}
END
```

E.12 Модули InventoryAndReadObjectsModules (инвентаризировать и считать объекты)

Модули InventoryAndReadObjectsModules (инвентаризировать и считать объекты) состоят из модуля команды (commandModule) и связанного с ним модуля ответа (responseModule), которые дают указания устройству опроса на идентификацию определенного набора радиочастотных меток, присутствующих в его рабочем поле, и возвращение определенных данных из каждой отдельной радиочастотной метки. Функция инвентаризации ограничена с помощью использования идентификатора AFI (applicationFamilyId) и метода идентификации (identifyMethod), как описано в E.4.

Аргумент команды objectIdList (список идентификаторов объектов) состоит из общего списка идентификаторов объектов (objectIds), которые подлежат считыванию из каждой радиочастотной метки, определенной критерием выбора. Если идентификатор objectIdList имеет нулевое значение, то устройство опроса должно получить все идентификаторы объектов и связанные с ними данные из каждой радиочастотной метки. Ответ радиочастотной метки с любым идентификатором tagId будет состоять из последовательности, состоящей из идентификатора объекта (objectId), объекта (object), параметра уплотнения (compactParameter), статуса блокировки (lockStatus).

Абстрактный синтаксис на языке ACH.1 для модулей InventoryAndReadObjectsModules (инвентаризировать и считать объекты) представлен в следующем примере:

```
-- Inventory and Read Objects (инвентаризировать и считать объекты)
-- InventoryAndReadObjectsCommand (команда «инвентаризировать и
-- считать объекты») дает указание устройству опроса на инвентаризацию
-- (то есть идентификацию) всех радиочастотных меток, присутствующих в
-- рабочей зоне, и считывание объектов, указанных в списке
-- идентификаторов объектов для каждой инвентаризируемой
-- радиочастотной метки. Любая радиочастотная метка идентифицируется
-- своим идентификатором TagId.
```

```
-- Если список идентификаторов объектов (objectIdList) имеет значение
-- NULL, то устройство опроса должно извлечь все объекты
-- инвентаризируемых радиочастотных меток.
-- Если присутствует дубликат идентификатора объекта, устройство
-- опроса должно вернуть в списке объектов его как блочный экземпляр
-- идентификатора объекта, объект и соответствующие параметры.
InventoryAndReadObjectsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126)
inventoryAndReadObjects(11)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
InventoryAndReadObjectsCommand ::= SEQUENCE {
     applicationFamilyId    ApplicationFamilyId,
     identifyMethod         INTEGER {
                              inventoryAllTags (0),
                              inventoryAtLeast (1),
                              inventoryNoMoreThan (2),
                              inventoryExactly (3)
                              }(0..15),
      numberOfTags           INTEGER (0..65535),
      objectIdList           SEQUENCE OF ObjectId
       }
-- Идентификатор ApplicationFamilyId отделяет принципиально разные
-- типы данных приложений и, возможно, конкретные идентификаторы
-- объектов (objectIds).
-- Указание шестнадцатеричного значения xx (где x - ненулевое
-- значение) позволяет выбрать только радиочастотные метки с требуемым
-- содержанием данных.
-- Указание шестнадцатеричного значения 00 позволяет выбрать все
-- радиочастотные метки; это может быть подходящим действием для
-- полной инвентаризации.
-- Указание шестнадцатеричного значения 0x или x0 (где x - ненулевое
-- значение) нелогично, поскольку радиочастотные метки из разных
-- приложений, а значение x имеет различный смысл.
-- Если для аргумента identifyMethod (метод идентификации) установлено
-- значение inventoryAllTags (инвентаризация всех меток), устройство
-- опроса должно выполнить полную инвентаризацию всех радиочастотных
-- меток, присутствующих в его рабочем поле. Значение numberOfTags
-- (число радиочастотных меток) не имеет к этому отношения и должно
-- быть установлено в нуль.
-- Если для аргумента identifyMethod (метод идентификации) установлено
-- значение inventoryAtLeast (инвентаризация не менее, чем),
-- устройство опроса должно выполнить инвентаризацию радиочастотных
-- меток, присутствующих в его рабочем поле, и должно вернуть ответ
-- только после того, как идентифицирует число радиочастотных меток,
-- равное заданному numberOfTags (число радиочастотных меток). Если
-- для numberOfTags установлено значение 1, устройство опроса будет
-- ожидать до обнаружения первой радиочастотной метки. Это механизм
-- ожидания входа радиочастотной метки в поле устройства опроса.
-- Если для numberOfTags установлено значение более 1, устройство
-- опроса будет ожидать, пока будет обнаружено указанное число
-- радиочастотных меток. Это может привести к неопределенному ожиданию
-- ответа от устройства опроса. Ответственность за использование этой
-- возможности лежит на приложении.
-- Если для аргумента identifyMethod (метод идентификации) установлено
-- значение inventoryNoMoreThan (инвентаризация не более, чем)
-- устройство опроса должно инициировать инвентаризацию радиочастотных
-- меток, присутствующих в его рабочем поле, и вернуть ответ с числом
```

```
-- радиочастотных меток, меньшим или равным заданному значению
-- numberOfTags. Устройство опроса может прервать процесс
-- инвентаризации, когда будет достигнуто заданное значение
-- numberOfTags или может продолжить процесс инвентаризации до тех
-- пор, пока все радиочастотные метки не будут считаны.
--   Примечание - Это может быть ограничено радиоинтерфейсом и
--   антиколлизионным механизмом.
ApplicationFamilyId : := SEQUENCE {
     applicationFamily     INTEGER(0..15),
     applicationSubFamily  INTEGER(0..15)
     }
     -- Кодовые значения определены в команде ConfigureAfiCommand
     -- (сконфигулировать идентификатор AFI).

ObjectId : := OBJECT IDENTIFIER -- Полное значение идентификатора
                                -- объекта

END
InventoryAndReadObjectsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127)
inventoryAndReadObjects(11)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
InventoryAndReadObjectsResponse : :=   SEQUENCE {
    completionCode      INTEGER {
                                noError (0),
                                objectsNotRead(16),
                                failedToReadMinimumNumberOfTags(23),
                                -- например, это может быть связано с
                                -- тайм-аутом
                                failedToReadExactNumberOfTags(24),
                                -- например, это может быть связано
                                -- с тайм-аутом
                                executionError(255)
                                },
    executionCode         INTEGER,
                          -- см. 9.4 и примечания к этому синтаксису
                          -- для полного списка значений executionCodes
    numberOfTagsFound     INTEGER (1..65535),
    tagIdAndObjects       SEQUENCE {
      tagId               OCTET STRING(SIZE(0..255)),
                          -- см. 7.2.1 для подробного описания
      objects             SEQUENCE OF SEQUENCE {
         objectId         OBJECT IDENTIFIER,
         object           OCTET STRING,
         compactParameter     INTEGER {
                              applicatioDefined(0),
                              -- Объект не был первоначально
                              -- закодирован по правилам уплотнения
                              -- данных ИСО/МЭК 15962 и в таком виде
                              -- передан от приложения-источника
                              -- и может потребовать дополнительной
                              -- обработки принимающим приложением.
                              utf8Data(2),
                              -- Данные были внешне преобразованы из
                              -- 16-битового кодового набора знаков в
                              -- строку знаков UTF-8. Объект должен
                              -- быть обработан с помощью внешнего
                              -- устройство декодирования знаков
```

```
                                -- UTF-8.
                                de-compactedData(15)
                                -- объект был первоначально уплотнен
                                -- по правилам ИСО/МЭК 15962,
                                -- разуплотнен при считывании и
                                -- восстановлен в своем первоначальном
                                -- формате.
                                }(0..15),
     lockStatus         BOOLEAN
                        -- если TRUE, объект заблокирован
               }
          }
    }
END
```

E.13 Модули EraseMemoryModules (очистить память)

Модули EraseMemoryModules (очистить память) состоят из модуля команды (commandModule) и связанного с ними модуля ответа (responseModule), которые дают указание устройству опроса на повторное обнуление всей карты логической памяти указанной радиочастотной метки. Он включает в себя directory (каталог), если тот определен как accessMethod (метод доступа). Если ни один из блоков не заблокирован, то он должен привести к удалению всех идентификаторов объектов (objectIds), объектов (objects) и связанных с ними прекурсоров (precursors). Если какой-либо блок заблокирован, то будет возвращен Completion-Code (код завершения): blocksLocked (блоки заблокированы). Объекты, которые остаются в радиочастотной метке, могут быть идентифицированы с помощью последующего использования модуля ReadObjectIdsModule (считать идентификаторы объектов) (см. E.9). Для каждой команды необходимо запрограммировать только одну радиочастотную метку, чтобы гарантировать надежность процесса очистки памяти.

Абстрактный синтаксис языка ACH.1 для модуля EraseMemoryModules (очистить память) приведен в следующем примере:

```
-- Erase Memory (очистить память)
-- EraseMemoryCommand (команда «очистить память») дает указание
-- устройству считывания сбросить на нуль всю карту логической памяти
-- радиочастотной метки, определенной идентификатором TagId.
-- Это приводит к удалению всех идентификаторов объектов
-- (ObjectIds), объектов (Objects) и связанных с ними параметров.
-- Объекты, которые заблокированы, не могут быть удалены. Если
-- некоторые объекты не могут быть удалены, поскольку они
-- заблокированы, устройство опроса должно вернуть соответствующий
-- Completion-Code (код завершения). Оставшиеся объекты могут быть
-- идентифицированы с помощью команды ReadObjectIdsCommand (считать
-- идентификаторы объектов).

EraseMemoryCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) eraseMemory (12)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
    TagId: :=        OCTET STRING(SIZE(0..255))
                     -- См. 7.2.1 для подробного описания
END

EraseMemoryResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) eraseMemory(12)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN

EraseMemoryResponse : :=  SEQUENCE {
    completionCode      INTEGER {
                         noError(0),
                         tagIdNotFound(8),
                         blocksLocked(17),
```

```
                        eraseIncomplete(18),
                        -- Процесс не завершен по неустановленной
                        -- причине, можно повторно вызвать команду
                        -- для завершения процесса
                        executionError(255)
                        },
    executionCode       INTEGER
                        -- См. 9.4 и примечания к этому синтаксису для
                        -- полного списка значений executionCodes
                        }
END
```

Е.14 Модули GetApplication-basedSystemInformationModules (получить системную информацию на базе приложения)

Модули GetApplication-basedSystemInformationModule (получить системную информацию на базе приложения) состоят из модуля команды (commandModule) и связанного с ним модуля ответа (responseModule), которые дают указание устройству опроса на считывание системной информации и возвращение тех аргументов, которые имеют отношение к приложению, а именно идентификатор семейства приложений (applicationFamilyId) и формат хранения (storageFormat).

Абстрактный синтаксис на языке АСН.1 для модулей GetApplication-basedSystemInformationModules (получить системную информацию на базе приложения) приведен в следующем примере:

```
-- GetApplication-basedSystemInformationModule (получить системную
-- информацию на базе приложения)
-- GetApp-basedSystemInfoCommand (команда «получить системную
-- информацию на базе приложения») дает указание устройству опроса на
-- считывание системной информации из радиочастотной метки с указанным
-- идентификатором TagId, а также абстрактных параметров,относящихся к
-- данным приложения: идентификатора applicationFamilyId и формата
-- хранения (storageFormat).
GetApp-basedSystemInfoCommand
{iso(1) standard 0) rfid-data-protocol(15961) commandModules(126)
getApp-basedSystemInfo(13)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
     TagId: :=   OCTET STRING(SIZE(0..255))
                 -- См. 7.2.1 для подробного описания

END
GetApp-basedSystemInfoResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127)
getApp-basedSystemInfo(13)}
DEFINITIONS
EXPLICIT TAGS : :=
BEGIN
GetApp-basedSystemInfoResponse ::=  SEQUENCE {
    completionCode            INTEGER {
                              noError(0),
                              tagIdNotFound(8),
                              systemInfoNotRead(20),
                              executionError(255)
                              },
    executionCode             INTEGER,
                              -- См. 9.4 и примечания к этому синтаксису для получения
                              -- полного списка кодовых значений executionCodes
    applicationFamilyId       ApplicationFamilyId,
    storageFormat             StorageFormat
     }
```

```
ApplicationFamilyId : := SEQUENCE {
    applicationFamily            INTEGER (0..15),
    applicationSubFamily         INTEGER (0..15)
    }
    -- Кодовые значения определяются в ConfigureAfiCommand
    -- (сконфигулировать идентификатор AFI).

StorageFormat : := SEQUENCE {
    accessMethod          INTEGER (0..3),
    dataFormat            INTEGER (0..31)
    }
    -- Кодовые значения определяются в команде сконфигурировать формат
    -- хранения (ConfigureStorageFormatCommand).
END
```

E.15 Модули AddMultipleObjectsModules (добавить блочные объекты)

Модули AddMultipleObjectsModules (добавить блочные объекты) включают в себя модуль команды (commandModule) и связанный модуль ответа (responseModule), и дают указание устройству опроса на запись набора объектов (objects), их идентификаторов (objectIds) и связанных параметров в карту логической памяти радиочастотной метки. Аргументы команды могут использоваться для индивидуального блокирования идентификатора объекта, объекта и связанных с ним параметров, и для проверки того, что идентификатор объекта еще не закодирован в радиочастотной метке. Для каждой команды необходимо запрограммировать только одну радиочастотную метку, чтобы обеспечить надежный процесс записи.

Абстрактный синтаксис на языке АСН.1 для модулей AddMultipleObjectsModules (добавить блочные объекты) представлен в следующем примере:

```
-- Add Multiple Objects (добавить блочные объекты)
-- AddMultipleObjectsCommand (команда «добавить блочные объекты») дает
-- указание устройству опроса на запись последовательности объектов,
-- их идентификаторов ObjectIds и связанных с ними параметров в карту
-- логическую памяти радиочастотной метки.
--   Примечание - Также существует команда AddSingleObjectCommand
--  (добавить единичный объект).
-- Если checkDuplicate flag (флаг проверки дубликата) установлен в
-- значение TRUE (ИСТИНА), устройство опроса прежде чем добавить
-- объекты, должно проверить, что объекта с тем же идентификатором не
-- существует. Если такой объект существует, устройство опроса не
-- должно выполнять функцию Add Object (добавить объект) и должно
-- вернуть соответствующее значение кода завершения (Completion Code).
-- Если Lock flag (флаг блокировки) установлен в значение TRUE (ИСТИНА),
-- устройство опроса должно заблокировать идентификатор объекта,
-- объект, его схему уплотнения и связанные с ним параметры в карте
-- логической памяти радиочастотной метки.
AddMultipleObjectsCommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126)
addMultipleObjects(14)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN
AddMultipleObjectsCommand ::= SEQUENCE {
    tagId               OCTET STRING(SIZE (0..255)),
                        -- См. 7.2.1 для подробного описания
    addObjectsList      SEQUENCE OF SEQUENCE{
        objectId            OBJECT IDENTIFIER,-- значение полного
                            -- идентификатора объекта
        avoidDuplicate      BOOLEAN,
                            -- Если установлено TRUE (ИСТИНА),
                            -- проверяется наличие дубликатов
                            -- идентификаторов объектов (objectId)
        object        OCTET STRING,
```

```
        compactParameter    INTEGER {
                              applicationDefined(0),
                              -- Объект не должен быть обработан по
                              -- правилам уплотнения данных
                              -- ИСО/МЭК 15962 и остается неизменным
                              compact(1),
                              -- Уплотнение объекта эффективно
                              -- настолько, насколько это определено
                              -- правилами уплотнения ИСО/МЭК 15962
                              utf8Data(2)
                              -- Данные были внешне преобразованы из
                              -- 16-битового кодового набора знаков в
                              -- строку знаков UTF-8. Объект не должен
                              -- быть обработан по правилам уплотнения
                              -- данных ИСО/МЭК 15962 и остается
                              -- неизменным
                              }(0..15),
    objectLock      BOOLEAN
                    -- Если TRUE (ИСТИНА), то устройство опроса
                    -- должно заблокировать объект, его схему
                    -- уплотнения и другие параметры в логической
                    -- карте памяти
        }
    }

END
AddMultipleObjectsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127)
addMultipleObjects(14)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

AddMultipleObjectsResponse ::= SEQUENCE {
      tagWriteResponse      SEQUENCE OF SEQUENCE{
          objectId             OBJECT IDENTIFIER,-- Full OID value
          completionCode       INTEGER{
                               noError(0),
                               tagIdNotFound(8),
                               objectNotAdded(9),
                               duplicateObject(10),
                               objectAddedButNotLocked(11),
                               executionError(255)
}
          },
      executionCode        INTEGER
                           -- См. 9.4 и примечания к этому синтаксису
                           -- для полного списка значений executionCodes
    }
END
```

E.16 Модули ReadMultipleObjectsModules (считать блочные объекты)

Модули ReadMultipleObjectsModules (считать блочные объекты) состоят из модуля команды (commandModule) и связанного с ними модуля ответа (responseModule), которые дают указание устройству опроса на считывание набора идентификаторов объектов (objectIds), объектов (objects) и связанных с ними параметров из карты логической памяти радиочастотной метки. Аргументы команды могут использоваться для проверки того, что идентификатор объекта не дублируется в радиочастотной метке. Только одна радиочастотная метка должна быть запрограммирована для каждой команды, чтобы гарантировать надежность процесса считывания.

Абстрактный синтаксис на языке АСН.1 для модулей ReadMultipleObjectsModules (считать блочные объекты) приведен в следующем примере:

```
-- Read Multiple Objects (считать блочные объекты)
-- ReadMultipleObjectsCommand (команда «считать блочные объекты») дает
-- указание устройству опроса на считывание объектов, определенных
-- идентификаторами объектов (Object Identifiers) для радиочастотной
-- метки, заданной идентификатором TagId.
--   Примечание - Существует также команда ReadSingleObjectCommand
--   (считать одиночный объект).
-- Если checkDuplicate flag (флаг проверки дубликата) имеет значение
-- FALSE (ЛОЖЬ), устройство опроса должно вернуть первый обнаруженный
-- объект, имеющий запрошенный идентификатор объекта без проверки
-- дубликатов.
-- Если checkDuplicate flag (флаг проверки дубликата) имеет значение
-- TRUE (ИСТИНА), устройство опроса должно проверить наличие дубликатов
-- объектов с запрошенным идентификатором объекта (OID). Если
-- присутствует более одного объекта с запрошенным идентификатором OID,
-- устройство опроса должно вернуть первый обнаруженный объект,
-- имеющий указанный идентификатор объекта и указать на присутствие
-- дубликатов с соответствующим значением Completion code.
ReadMultipleObjectsCommand
{iso(1) standard(0) rfid-data-protocol 15961) commandModules 126)
readMultipleObjects(15)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN
ReadMultipleObjectsCommand ::= SEQUENCE {
    tagId                OCTET STRING(SIZE(0..255)),
                         -- См. 7.2.1 для подробного описания
    readObjectList       SEQUENCE OF SEQUENCE{
        objectId            OBJECT IDENTIFIER, -- Полное значение
                            -- идентификатора OID
        checkDuplicate      BOOLEAN
                    -- Если установлено TRUE, то устройство
                    -- опроса должно проверить наличие только
                    -- одного вхождения идентификатора
                    -- объекта (ObjectId)
        }
    }

END
ReadMultipleObjectsResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127)
readMultipleObjects(15)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ReadMultipleObjectsResponse ::= SEQUENCE {
      tagReadResponse         SEQUENCE OF SEQUENCE{
           objectId           OBJECT IDENTIFIER,
           object             OCTET STRING,
           compactParameter   INTEGER {
                              applicationDefined(0),
                              -- Объект был первоначально закодирован
                              -- по правилам уплотнения данных
                              -- ИСО/МЭК 15962 и передан из
                              -- приложения-источника и может
                              -- потребовать дополнительной обработки
                              -- в принимающем приложении.
                             utf8Data(2),
                             -- Данные были внешне преобразованы из
                             -- 16-битового кодового знаков набора в
```

```
                              -- строку знаков UTF-8. Объект должен
                              -- быть обработан с помощью внешнего
                              -- устройства декодирования знаков
                              -- UTF-8.
                              de-compactedData(15)
                              -- Объект был первоначально уплотнен
                              -- по правилам ИСО/МЭК 15962 и
                              -- разуплотнен при считывании и
                              -- восстановлен в первоначальном
                              -- формате.
                              }(0..15),
lockStatus          BOOLEAN,
                    -- Если TRUE, то объект заблокирован
completionCode      INTEGER {
                                    noError(0),
                                    tagIdNotFound(8),
                                    duplicateObject(10),
                                    objectIdNotFound(13),
                                    objectNotRead(15),
                                    executionError(255)
                                    }
   },
   executionCode        INTEGER
                        -- См. 9.4 и примечания к этому синтаксису
                        -- для полного списка значений executionCodes
    }
END
```

E.17 Модули ReadFirstObjectModules (считать первый объект)

Модули ReadFirstObjectModules (считать первый объект) состоят из модуля команды (commandModule) и связанного с ним модуля ответа (responseModule), которые дают указание устройству опроса на считывание объекта в первой позиции в радиочастотной метке, возвращая значение его идентификатора объекта (objectId), объекта (object) и связанных с ним параметров из карты логической памяти радиочастотной метки. Только одна радиочастотная метка должна быть запрограммирована для каждой команды, чтобы гарантировать надежность процесса считывания.

Эффективный процесс может быть достигнут благодаря возможности передавать только соответствующие блоки данных по радиоинтерфейсу. Требуется, чтобы приложение предоставляло следующее:

- некоторую степень уверенности в том, что ожидаемый объект закодирован в конкретной позиции, что может быть достигнуто правилами в стандарте по применению;
- известную или максимальную уплотненную длину ожидаемого объекта. Это используется для определения числа передаваемых октетов и получения ожидаемого возвращаемого объекта. Величина уплотненной длины может быть определена с помощью процедуры, описанной ниже;
- значение ожидаемого идентификактора объекта (objectId), чтобы позволить процессору обработки протокола данных (Data Protocol Processor) преобразовать его в закодированные октеты и добавить его и другие закодированные параметры к величине уплотненной длине.

Соответствующие схемы уплотнения и знаки, которые могут быть уплотнены, определены в таблице Е.1.

Таблица Е.1 — Схемы уплотнения и знаки

Схемы уплотнения	Число знаков	Поддерживаемые знаки
Числовая (4-бита)	10	0..9
5-битовая	31	A..Z [\] ^_
6-битовая	64	Как в числовой + 5-битовой схеме уплотнения и следующие знаки: ПРОБЕЛ ! “ # $ % & ‘ () * + , - : ; < = > ? @
7-битовая	128	Все знаки по ИСО/МЭК 646*, включая управляющие знаки
8-битовая	256	Все знаки по ИСО/МЭК 8859-1**, включая управляющие знаки

* *См. [1].*

** *См. [4].*

В качестве примера использования рассмотрен стандарт по применению, требующий, чтобы уникальный идентификатор предмета (Unique Item Identifier) для транспортируемых единиц по ИСО/МЭК 15459-1* находился в первой позиции. Максимальная длина данного идентификатора составляет 35 алфавитно-цифровых знаков. Предположим, что конкретный пункт сбора данных обрабатывает транспортируемые единицы из различных пунктов и не имеет никаких предварительных знаний о длине отдельных кодовых значений, поэтому в худшем случае должны быть соблюдены условия таблицы Е.1, предоставляющие список знаков, подлежащих кодированию схемами уплотнения по ИСО/МЭК 15962. Алфавитно-цифровые знаки, указанные в ИСО/МЭК 15459-1*, могут быть уплотнены с помощью 6-битовой схемы. Таким образом, максимальная длина закодированного объекта составляет (35 × 6/8 = 26,25) 27 октетов, поскольку для надлежащего кодирования значение должно быть округлено. Процессор обработки протокола данных (Data Protocol Processor) добавляет закодированную длину идентификатора объекта (objectId) и связанные параметры для определения числа блоков, подлежащих передаче.

Если известна длина и структура объекта, например 20-значное кодовое значение, таблица демонстрирует, что объект может быть уплотнен с использованием схемы 4 бита на одну цифру. Таким образом, полная длина уплотненного объекта составляет 10 октетов. Хотя целочисленное уплотнение может быть вызвано для создания более короткого закодированного объекта, вычисление длины на основе более простого числового уплотнения охватывает все случаи для числовых уникальных идентификаторов объекта (Unique item Identifiers).

Основная цель состоит в том, чтобы свести к минимуму передачу по радиоинтерфейсу с высокой степенью уверенности в том, что объект возвращен. Можно уточнить процесс для определения числа октетов. Длина ожидаемого объекта может быть более точно рассчитана путем моделирования процесса уплотнения с использованием образцов реальных возвращенных объектов.

Абстрактный синтаксис на языке АСН.1 для модулей ReadFirstObjectModules (считать первый объект) представлен в следующем примере:

```
-- Read First Object (считать первый объект)
-- ReadFirstCommand (команда «считать первый объект») дает указание
-- устройству опроса на считывание объекта, определенного его позицией,
-- хранящейся в радиочастотной метке.

-- Данная команда может иметь параметры в радиоинтерфейсе, который может
-- работать быстрее, чем считывание одного именованного объекта.
-- Приложение может выбрать наиболее часто используемый объект,
-- хранящийся первым в радиочастотной метке.

ReadFirstObjectcommand
{iso(1) standard(0) rfid-data-protocol(15961) commandModules(126) readFirstObject(16)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN

ReadFirstObjectCommand ::= SEQUENCE {

    TagId               OCTET STRING(SIZE(0..255)),
                        -- см. 7.2.1 для полной спецификации

    objectId            OBJECT IDENTIFIER, -- Это ожидаемый идентификатор
                        -- объекта (objectId). Это значение используется
                        -- процессами по ИСО/МЭК 15962, чтобы определить
                        -- число блоков, которые должны быть переданы.
                        -- Это не используется как часть самого процесса
                        -- поиска.
    maxAppLength        INTEGER(1..65535)
                        -- Это значение задается приложением для
                        -- представления максимальной уплотненной
                        -- длины ожидаемого объекта
    }

END

ReadFirstObjectResponse
{iso(1) standard(0) rfid-data-protocol(15961) responseModules(127) readFirstObject(16)}
DEFINITIONS
EXPLICIT TAGS ::=
BEGIN
```

* *См.* [*24*].

```
ReadFirstObjectResponse ::= SEQUENCE {
    ObjectId                    OBJECT IDENTIFIER,
                                -- Это текущий идентификатор объекта
                                -- (ObjectId) как обнаружено в первой
                                -- позиции
    object                      OCTET STRING,
    compactParameter            INTEGER {
                          applicationDefined(0),
                          -- Объект не был первоначально закодирован
                          -- по правилам уплотнения данных ИСО/МЭК
                          -- 15962 и в таком виде передан от
                          -- приложения-источника, может потребовать
                          -- дополнительной обработки принимающим
                          -- приложением.
                          utf8Data(2),
                          -- Данные были внешне преобразованы из
                          -- 16-битового кодового набора знаков в
                          -- строку знаков UTF-8. Объект должен быть
                          -- обработан с помощью внешнего устройства
                          -- декодирования знаков UTF-8.
                          de-compactedData(15)
                          -- Объект был первоначально закодирован по
                          -- правилам уплотнения данных ИСО/МЭК 15962
                          --  и разуплотнен при операции считывания и
                          -- восстановлен в первоначальном формате.
                          }(0..15),
     lockStatus          BOOLEAN,
                         -- Если TRUE, то объект заблокирован
     completionCode      INTEGER {
                            noError(0),
                            tagIdNotFound(8),
                            objectIdNotFound(13),
                            objectNotRead(15),
                            executionError(255)
                         },
     executionCode       INTEGER
                         -- См. 9.4 и примечания к этому синтаксису для
                         -- полного списка значений executionCodes
     }
END
```

Приложение F
(справочное)

Пример кодирования передачи по ИСО/МЭК 15961:2004

F.1 Общие положения

Данное приложение иллюстрирует основные правила кодирования, приведенные в ИСО/МЭК 15961:2004*, с демонстрацией представления в октетах команды (гипотетической) и ответа AddMultipleObjects (добавить блочные объекты).

F.2 Функциональное описание команды

Функция иллюстрируемой команды — это запись двух объектов данных в радиочастотную метку, которая имеет уникальный идентификатор объекта (Unique Item Identifier - UII) C7 37 79 C2 B7 A3 DB EF. Прочие подробные сведения:

- 1-й идентификатор объекта «1 0 15961 10 30», с объектом «ABC123456», который должен быть заблокирован;
- 2-й идентификатор объекта «1 0 15961 10 17», с объектом «50», который не должен быть заблокирован;
- отсутствует требование избегать дублирования данных, которые уже закодированы в радиочастотной метке;
- оба объекта должны быть уплотнены.

F.3 Абстрактный синтаксис команды AddMultipleObjects (добавить блочные объекты)

Приведен в приложении Е.15 приложения Е.

F.4 Команда AddMultipleObjects (добавить блочные объекты) со значениями данных

```
AddMultipleObjectsCommand        -- {1 0 15961 126 14}
    ::=     {                    {
        tagId                     C7 37 79 C2 B7 A3 DB EF 16
        addObjectsList            {
        --  первый объект
        {
        objectId                  {1 0 15961 10 30},
        avoidDuplicate            FALSE,
        object                    “ABC123456”,
        compactParameter          compact(1),
        objectLock                TRUE
        },
        -- второй объект {
        objectId                  {1 0 15961 10 17},
        avoidDuplicate            FALSE,
        object                    “50”,
        compactParameter          compact(1),
        objectLock                FALSE
        }
    }
}
```

Примечание — Значения идентификатора objectId представлено как OBJECT IDENTIFIERS. Значения объекта представлены в кавычках (“ ”) как печатные знаки.

F.5 Пример команды кодирования передачи

Кодирование передачи в октетах приведенного ранее значения команды (после применения правил кодирования передачи по ИСО/МЭК 15961:2004*) приведено ниже в табличной форме, причем каждая строка таблицы представляет собой одну из строк, приведенных в F.4. Значения идентификаторов типа (type), длины (length) и содержимого (content) показаны в шестнадцатеричном виде, двумя шестнадцатеричными разрядами на октет.

Абстрактный синтаксис	Тип[1] (Type)	Длина (Length)	Значение (Value)
AddMultipleObjectsCommand (добавить блочные объекты) [2]	06	05	28 FC 59 7E 0E
SEQUENCE (последовательность) [3]	30	3F	
TagId (идентификатор TagId)	04	08	C7 37 79 C2 B7 A3 DB EF

* *См.* [*8*].

Абстрактный синтаксис	Тип[1] (Type)	Длина (Length)	Значение (Value)
SEQUENCE (AddObjectsList) [4,5] (список добавленных объектов)	30	33	
SEQUENCE (последовательность) [3]	30	1B	
objectId (идентификатор объекта)	06	05	28 FC 59 0A 1E
AvoidDuplicate (избежание дублирования) [6]	01	01	00
Object (объект)	04	09	41 42 43 31 32 33 34 35 36
CompactParameter (параметр уплотнения)[7]	02	01	01
ObjectLock (блокировка объекта) [6]	01	01	FF
SEQUENCE (последовательность) [3]	30	14	
objectId (идентификатор объекта)	06	05	28 FC 59 0A 11
AvoidDuplicate (избежание дублирования) [6]	01	01	00
Object (объект)	04	02	35 30
CompactParameter (параметр уплотнения)	02	01	01
ObjectLock (блокировка объекта) [6]	01	01	00

Объяснение конкретных закодированных значений приведено ниже, где порядковый номер соотносится с номером ссылки в строках табличной формы.

1 Значения в столбце «Тип» получают на основе правил 6.2.2 ИСО/МЭК 15961:2004*.

2 Модуль OBJECT IDENTIFIER кодируют с использованием тех же правил, что и любой идентификатор объекта.

3 Как указано в 6.2.9 ИСО/МЭК 15961:2004*, SEQUENCE (последовательность) не имеет связанного значения.

4 Как указано в 6.2.10 ИСО/МЭК 15961:2004*, аргумент addObjectsList (список добавленных объектов) определяет SEQUENCE OF (последовательность из), которая не имеет связанного значения.

5 Поскольку существует два набора objectId (идентификатор объекта) и связанных с ними данных, длина всего вложенного закодированного значения должна быть закодирована в строке.

6 Аргументы avoidDuplicate (избежание дублирования) и objectLock (блокировка объекта) имеют BOOLEAN (ЛОГИЧЕСКОЕ ЗНАЧЕНИЕ). Закодированное значение для BOOLEAN = FALSE (ЛОЖЬ) должно быть 00. Закодированное значение для BOOLEAN = TRUE (ИСТИНА) может быть любым значением, отличным от 00.

7 Как определено в 7.3.3 ИСО/МЭК 15961:2004*, аргумент compactParameter (параметр уплотнения) кодируется как INTEGER (ЦЕЛОЧИСЛЕННОЕ ЗНАЧЕНИЕ).

Полное кодирование передачи команды в виде потока октетов:

06 05 28 FC 59 7E 0E 30 3F 04 08 C7 37 79 C2 B7 A3 DB EF 30 33 30 1B 06 05 28 FC 59 0A 1E 01 01 00 04 09 41 42 43 31 32 33 34 35 36 02 01 01 01 01 FF 30 14 06 05 28 FC 59 0A 11 01 01 00 04 02 35 30 02 01 01 01 01 00.

F.6 Функциональное описание ответа на команду

В этом примере предполагают, что радиочастотная метка была обнаружена, и оба набора идентификатор объекта (Object Identifier) и объект (object), были надлежащим образом добавлены в радиочастотную метку с использованием всех соответствующих процессов по ИСО/МЭК 15962, с одним существенным исключением: первый объект не мог быть заблокирован в соответствии с запросом команды. Когда команда была успешно выполнена с точки зрения системных коммуникаций, возвращаемое значение Execution-Code (кода выполнения) — “0 noError” (0 нет ошибки).

F.7 Абстрактный синтаксис ответа на команду AddMultipleObjects response (добавить блочные объекты)

Приведен в Е.15 приложения Е.

F.8 Ответ на команду AddMultipleObjects (добавить блочные объекты) со значениями данных

```
AddMultipleObjectsResponse                    -- { 1 0 15961 127 14 }
        ::=                                   {
    tagWriteResponse          {
        -- ответ на команду добавить 1-й объект
```

* *См.* [*8*].

```
            {
            objectId                                { 1 0 15961 10 30 },
            completionCode                          11
            },
            -- ответ на команду добавить 2-й объект
             {
            objectId                                { 1 0 15961 10 17 },
            completionCode                          0
            }
    executionCode 0
}
```

F.9 Кодирование передаваемых данных для примера ответа на команду

Кодирование передаваемых данных в октетах приведенного выше значения ответа на команду (после применения базовых правил кодирования передачи, определенных в настоящем стандарте) приведено ниже в табличной форме, причем каждая строка таблицы представляет собой одну из строк, приведенных в F.8. Значения идентификаторов типа (type), длины (length) и содержимого (content) показаны в шестнадцатеричном виде, две шестнадцатеричные цифры на октет.

Абстрактный синтаксис	Тип (Type)	Длина (Length)	Значение (Value)
AddMultipleObjectsResponse (ответ на команду «добавить блочные объекты»)	06	05	28 FC 59 7F 0F
SEQUENCE (последовательность)	30	1D	
SEQUENCE OF (TagWriteResponse) [последовательность из (ответ на команду TagWriteResponse «записать радиочастотную метку»)]	30	18	
SEQUENCE (последовательность)	30	0A	
ObjectId (идентификатор объекта)	06	05	28 FC 59 0A 1E
CompletionCode (код завершения)	02	01	0B
SEQUENCE (последовательность)	30	0A	
ObjectId (идентификатор объекта)	06	05	28 FC 59 0A 11
CompletionCode (код завершения)	02	01	00
ExecutionCode (код выполнения)	02	01	00

Полное кодирование передачи ответа на команду в виде потока октетов имеет следующий вид:

06 05 28 FC 59 7F 0F 30 1D 30 18 30 0A 06 05 28 FC 59 0A 1E 02 01 0B 30 0A 06 05 28 FC 59 0A 11 02 01 00 02 01 00.

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным и межгосударственным стандартам

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального, межгосударственного стандарта
ISO/IEC 15961-3	IDT	ГОСТ Р ИСО/МЭК 15961-3—2021 «Информационные технологии. Протокол данных радиочастотной идентификации для управления предметами. Часть 3. Конструкции данных радиочастотной идентификации»
ISO/IEC 15962	—	*
ISO/IEC 19762	MOD	ГОСТ 30721—2020 (ISO/IEC 19762:2016) «Информационные технологии. Технологии автоматической идентификации и сбора данных (АИСД). Гармонизированный словарь»

* Соответствующий национальный, межгосударственный стандарт отсутствует. До его принятия рекомендуется использовать перевод на русский язык данного международного стандарта.

Примечание — В настоящей таблице использованы следующие условные обозначения степени соответствия стандартов:

IDT — идентичный стандарт;

MOD — модифицированный стандарт.

Библиография

[1] ISO/IEC 646 Information technology — ISO 7-bit coded character set for information interchange (Информационные технологии. 7-битовый набор кодированных знаков для обмена информацией)

[2] ISO/IEC 8824-1 Information technology — Abstract Syntax Notation One (ASN.1) — Specification of basic notation (equivalent to ITU-T Recommendation X.680) [Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации (эквивалентна Рекомендациям МСЭ-Т X.680)]*

[3] ISO/IEC 8825-1 Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) (equivalent to ITU-T Recommendation X.690) [Информационная технология. Правила кодирования АСН.1. Часть 1. Спецификация базовых (BER), канонических (CER) и отличительных (DER) правил кодирования (эквивалентна Рекомендациям X.690 МСЭ-Т)]**

[4] ISO/IEC 8859-1 Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1 (Информационные технологии. 8-битовые однобайтовые наборы кодированных графических знаков. Часть 1. Латинский алфавит № 1)

[5] ISO/IEC 9834-1 Information technology — Procedures for the operation of object identifier registration authorities — Part 1: General procedures and top arcs of the international object identifier tree (Информационные технологии. Процедуры для работы регистрационных органов по идентификации объекта. Часть 1. Общие процедуры и высшие разряды дерева идентификаторов объекта международного объекта)***

[6] ISO/IEC 10646 Information technology — Universal coded character set (UCS) [Информационные технологии. Универсальный набор кодированных знаков (UCS)]

[7] ISO/IEC 15434 Information technology — Automatic identification and data capture techniques — Syntax for high-capacity ADC media [Информационные технологии. Технологии автоматической идентификации и сбора данных. Синтаксис для средств автоматического сбора данных (ADC) большой емкости]*4

[8] ISO/IEC 15961:2004 Information technology — Radio frequency identification (RFID) for item management — Data protocol: application interface (Информационные технологии. Радиочастотная идентификация для управления предметами. Протокол данных: прикладной интерфейс)

[9] ISO/IEC 15961-2 Information technology — Radio frequency identification (RFID) for item management: Data protocol — Part 2: Registration of RFID data constructs (Информационные технологии. Радиочастотная идентификация для управления предметами: Протокол данных. Часть 2. Регистрация конструкций данных радиочастотной идентификации)*5

[10] ISO/IEC 18000-2 Information technology — Radio frequency identification for item management — Part 2: Parameters for air interface communications below 135 kHz (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 2. Параметры радиоинтерфейса для связи на частотах ниже 135 кГц)

[11] ISO/IEC 18000-3 Information technology — Radio frequency identification for item management — Part 3: Parameters for air interface communications at 13,56 MHz (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 3. Параметры радиоинтерфейса для связи на частоте 13,56 МГц)*6

---

* *Действует ГОСТ Р ИСО/МЭК 8824-1—2001 «Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации».*

** *Действует ГОСТ Р ИСО/МЭК 8825-1—2003 «Информационная технология. Правила кодирования АСН.1. Часть 1. Спецификация базовых (BER), канонических (CER) и отличительных (DER) правил кодирования».*

*** *Действует ГОСТ Р ИСО/МЭК 9834-1—2009 «Информационная технология. Взаимосвязь открытых систем. Процедуры действий уполномоченных по регистрации ВОС. Часть 1. Общие процедуры и верхние дуги дерева идентификатора объекта АСН.1».*

*4 *Действует ГОСТ 34731—2021 (ISO/IEC 15434:2019) «Информационные технологии. Технологии автоматической идентификации и сбора данных. Синтаксис носителей данных высокой емкости для автоматического сбора данных».*

*5 *Действует ГОСТ Р ИСО/МЭК 15961-2—2021 «Информационные технологии. Протокол данных радиочастотной идентификации для управления предметами. Часть 2. Регистрация конструкций данных радиочастотной идентификации».*

*6 *Действует ГОСТ Р 58666—2019 (ИСО/МЭК 18000-3:2010) «Информационные технологии. Идентификация радиочастотная для управления предметами. Параметры радиоинтерфейса для связи на частоте 13,56 МГц».*

[12] ISO/IEC 18000-4 Information technology — Radio frequency identification for item management — Part 4: Parameters for air interface communications at 2,45 GHz (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 4. Параметры радиоинтерфейса для связи на частоте 2,45 ГГц)

[13] ISO/IEC 18000-6 Information technology — Radio frequency identification for item management — Part 6: Parameters for air interface communications at 860 MHz to 960 MHz (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 6. Параметры радиоинтерфейса для связи на частотах от 860 МГц до 960 МГц)*

[14] ISO/IEC 18000-61 Information technology — Radio frequency identification for item management — Part 61: Parameters for air interface communications at 860 MHz to 960 MHz Type A (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 61. Параметры радиоинтерфейса для связи в диапазоне частот 860—960 МГц, тип А)

[15] ISO/IEC 18000-62 Information technology — Radio frequency identification for item management — Part 62: Parameters for air interface communications at 860 MHz to 960 MHz Type B (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 62. Параметры радиоинтерфейса для связи в диапазоне частот 860—960 МГц, тип В)**

[16] ISO/IEC 18000-63 Information technology — Radio frequency identification (RFID) for item management — Part 63: Parameters for air interface communications at 860 MHz to 960 MHz Type C (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 63. Параметры радиоинтерфейса для связи в диапазоне частот 860—960 МГц, тип С)***

[17] ISO/IEC 18000-64 Information technology — Radio frequency identification for item management — Part 64: Parameters for air interface communications at 860 MHz to 960 MHz Type D (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 64. Параметры радиоинтерфейса для связи в диапазоне частот 860—960 МГц, тип D)

[18] ISO/IEC 18000-7 Information technology — Radio frequency identification for item management — Part 7: Parameters for active air interface communications at 433 MHz (Информационные технологии. Радиочастотная идентификация для управления предметами. Часть 7. Параметры активного радиоинтерфейса для связи на частоте 433 МГц)

[19] ISO/IEC 24791-1 Information technology — Radio frequency identification (RFID) for item management — Software system infrastructure — Part 1: Architecture (Информационные технологии. Радиочастотная идентификация для управления предметами. Инфраструктура программного обеспечения системы. Часть 1. Архитектура)

[20] ISO/IEC 24791-2 Information technology — Radio frequency identification (RFID) for item management — Software system infrastructure — Part 2: Data management (Информационные технологии. Радиочастотная идентификация для управления предметами. Инфраструктура программного обеспечения системы. Часть 2. Управление данными)

[21] ISO/IEC 24791-5 Information technology — Radio frequency identification (RFID) for item management — Software system infrastructure — Part 5: Device interface (Информационные технологии. Радиочастотная идентификация для управления предметами. Инфраструктура программного обеспечения системы. Часть 5. Интерфейс устройства)

[22] GS1 EPC Tag Data Standard (TDS) [Стандарт данных радиочастотной метки GS1 EPC (СДРМ)]

[23] IETF RFC 3061 A URN Namespace of Object Identifiers, published by The Internet Engineering Task Force, part of The Internet Society (2001) [Пространство имен URN для идентификаторов объектов, опубликовано Рабочей группой инженерной поддержки сети Интернет, части сообщества Интернет (2001)]

---

* *Действует ГОСТ 34693.6—2020 (ISO/IEC 18000-6:2013) «Информационные технологии. Идентификация радиочастотная для управления предметами. Часть 6. Параметры радиоинтерфейса для диапазона частот 860—960 МГц. Общие требования».*

** *Действует ГОСТ Р ИСО/МЭК 18000-62—2014 «Информационные технологии. Идентификация радиочастотная для управления предметами. Часть 62. Параметры радиоинтерфейса для связи в диапазоне частот 860—960 МГц, тип В».*

*** *Действует ГОСТ Р 58701—2019 (ИСО/МЭК 18000-63:2015) «Информационные технологии. Идентификация радиочастотная для управления предметами. Параметры радиоинтерфейса для связи в диапазоне частот от 860 МГц до 960 МГц (Тип С)».*

[24] ISO/IEC 15459-1 Information technology — Automatic identification and data capture techniques — Unique identification — Part 1: Individual transport units (Информационные технологии. Технологии автоматической идентификации и сбора данных. Уникальная идентификация. Часть 1. Индивидуальные транспортируемые единицы) *

* *Действует ГОСТ ISO/IEC 15459-1—2016 «Информационные технологии. Технологии автоматической идентификации и сбора данных. Идентификация уникальная. Часть 1. Индивидуальные транспортируемые единицы».*

УДК 003.62:681.3.04:681.3.053:006.354 ОКС 35.040.50

Ключевые слова: радиочастотная идентификация для управления предметами, протокол данных, прикладной интерфейс

Редактор *Н.В. Таланова*
Технический редактор *В.Н. Прусакова*
Корректор *О.В. Лазарева*
Компьютерная верстка *Е.О. Асташина*

Сдано в набор 17.11.2023. Подписано в печать 12.12.2023. Формат 60×84⅛. Гарнитура Ариал.
Усл. печ. л. 14,88. Уч.-изд. л. 13,40.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Создано в единичном исполнении в ФГБУ «Институт стандартизации»
для комплектования Федерального информационного фонда стандартов,
117418 Москва, Нахимовский пр-т, д. 31, к. 2.

gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru