ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО/МЭК 8825-6— 2013

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

Правила кодирования АСН.1

Часть 6

Регистрация и применение команд кодирования PER

ISO/IEC 8825-6:2008
Information technology – ASN.1 encoding rules – Part 6: Registration and application of PER encoding instructions
(IDT)

Издание официальное

Москва
Стандартинформ
2014

Предисловие

1 ПОДГОТОВЛЕН Федеральным государственным унитарным предприятием Государственный научно-исследовательский и конструкторско-технологический институт «ТЕСТ» (ФГУП ГосНИИ «ТЕСТ») на основе собственного аутентичного перевода на русский язык международного стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 22 «Информационные технологии»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 6 сентября 2013 г. № 877-ст

4 Настоящий стандарт идентичен международному стандарту ИСО/МЭК 8825-6:2008 «Информационная технология. Правила кодирования АСН.1. Часть 6. Регистрация и применение команд кодирования PER» (ISO/IEC 8825-6:2008 «Information technology – ASN.1 encoding rules – Part 6: Registration and application of PER encoding instructions»).

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

5 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта установлены в ГОСТ Р 1.0—2012 (раздел 8). 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 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.ru)

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

Правила кодирования АСН.1.

Часть 6

Регистрация и применение команд кодирования PER

Information technology. ASN.1 encoding rules. Part 6.
Registration and application of PER encoding instructions

Дата введения — 2014-07-01

1 Область применения

В настоящем стандарте определены:

a) необходимая информация и формат, который должен быть использован для описания команд кодирования PER;

b) механизмы одобрения новых команд кодирования PER и работа службы регистрации (Registration Authority) команд кодирования PER;

c) средства сопоставления команды кодирования PER с типом АСН.1, используя префиксы типа и секции контроля кодирования.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие стандарты (идентичные рекомендации и международные стандарты):

– Рекомендация МСЭ-Т X.680 (2008) (ИСО/МЭК 8824-1:2008) Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации.

– Рекомендация МСЭ-Т X.691 (2008) (ИСО/МЭК 8825-2:2008) Информационная технология. Правила кодирования АСН.1. Часть 2. Спецификация правил уплотненного кодирования (PER).

Примечание – При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год Если заменен ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Определения

В настоящем стандарте использованы термины с соответствующими определениями по ИСО/МЭК 8824-1, а также следующие дополнительные термины с соответствующими определениями.

3.1 **связанные команды кодирования (для типа)**: Набор команд кодирования PER, связанных с типом.

Издание официальное

3.2 **заключительные команды кодирования (для типа)**: Набор команд кодирования PER, связанных с типом, как результат полной спецификации АСН.1, которые применяются при создании кодировок этого типа.

3.3 **идентификационное ключевое слово:** Слово или написанное через дефис слово, используемое для идентификации команды кодирования PER.

3.4 **наследованные команды кодирования**: Команды кодирования PER, которые связаны с типом, идентифицированным ссылкой типа.

3.5 **совместная группа из МСЭ-Т и ИСО/МЭК СТК 1 для АСН.1**: Группа, созданная в соответствии с рекомендацией МСЭ-Т А.23, приложение А, и распоряжениями ИСО/МЭК СТК 1, выпуск 5, версия 2.0, подпункт 2.6.4 и приложение К, пункт 8, для продвижения работы над объединенным текстом (Joint Text) по отношению к абстрактной синтаксической нотации версии один (АСН.1).

3.6 **команды кодирования PER (КК-ы PER)**: Нотация, используемая для изменения невыровненного PER кодирования типа (или компонента типа).

Примечание – Команды кодирования PER включаются либо в префикс типа PER (см. Рек. МСЭ-Т Х.680 ИСО/МЭК 8824-1, 31.3), либо в секцию контроля кодирования PER (см. Рек. МСЭ-Т Х.680 ИСО/МЭК 8824-1, пункт 54).

3.7 **предложение КК PER:** Предложение новой команды кодирования PER, которое приводит либо к состоянию REJECTED, либо к состоянию APPROVED.

3.8 **предложение изменения КК PER:** Предложение по удалению или изменению APPROVED команды кодирования PER, которое приводит либо к состоянию REJECTED, либо к состоянию APPROVED.

Примечание – Ожидается, что предложения изменения КК PER возникают редко и в течение рассмотрения потребуются для рассмотрений обратной совместимости.

3.9 **команды кодирования с префиксом**: Команды кодирования PER, которые присваиваются, используя префикс типа (см. приложение А и Рек. МСЭ-Т Х.680 ИСО/МЭК 8824-1, п. 31.3).

Примечание – Команды кодирования с префиксом можно удалить, заменить или добавить к связанным командам кодирования типа.

3.10 **соответствующая рабочая группа**: Рабочая группа МСЭ-Т, которая отвечает за совместную группу МСЭ-Т и ИСО/МЭК СТК 1 для АСН.1.

3.11 **соответствующий подкомитет**: Подкомитет ИСО/МЭК СТК1, который отвечает за совместную группу МСЭ-Т и ИСО/МЭК СТК 1 для АСН.1.

3.12 **целевые команды кодирования**: Команды кодирования PER, которые присваиваются, используя целевой список в секции контроля кодирования PER (см. приложение В и Рек. МСЭ-Т Х.680 ИСО/МЭК 8824-1, п. 54).

Примечание – Целевые команды кодирования можно удалить, заменить или добавить к связанным командам кодирования типа.

4 Сокращения

В настоящем стандарте применены следующие сокращения:
АСН.1 – Abstract Syntax Notation One (абстрактная синтаксическая нотация версии 1);
ECN – Encoding Control Notation (нотация контроля кодирования);
EI – Encoding Instruction (команда кодирования);
PER – Packed Encoding Rules (правила уплотненного кодирования).

5 Нотация

5.1 Настоящий стандарт ссылается и использует нотацию, описанную в Рек. МСЭ-Т Х.680 ИСО/МЭК 8824-1, п. 5, для определения синтаксиса команд кодирования PER через набор процедур.

5.2 Все лексические элементы, используемые в этих процедурах, определены Рек. МСЭ-Т Х.680 ИСО/МЭК 8824-1, п. 12.

5.3 В соответствии с Рек. МСЭ-Т Х.680 ИСО/МЭК 8824-1, 31.3.2, настоящий стандарт

определяет процедуру «EncodingInstruction» (см. 11.2).

Примечание – Лексические элементы «[» и «]» никогда не появляются в процедуре «EncodingInstruction».

5.4 В соответствии с Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1, п. 54.4, настоящий стандарт определяет процедуру «EncodingInstructionAssignmentList» (см. 12.1.2).

Примечание – Лексические элементы **END** и **ENCODING-CONTROL** никогда не появляются в процедуре «EncodingInstructionAssignmentList»

6 Информация, которая должна быть указана для описания команды кодирования PER

6.1 Описание команды кодирования PER должно содержать следующую информацию:

a) короткий информативный заголовок (такой как «Использование и размер поля длины»);

b) значение для идентификационного ключевого слова, которое должно быть отличным от любого используемого ранее для КК PER;

c) иллюстрацию синтаксиса КК, включенного в квадратные скобки («[» и «]»);

d) простое описание КК и ее цели;

e) полную спецификацию типов, к которым эта КК применяется и результат ее применения в каждом случае;

f) полную спецификацию процедуры «EIDetail» (см. 11.4) с семантикой, связанной с каждым элементом;

g) также:

1) поправку к определенному тексту в Рек. МСЭ-Т X.691 ИСО/МЭК 8825-2, которая должна быть применена к кодированию любого типа, к которому КК применяема (см. d) выше) и которая имеет эту КК в наборе заключительных команд кодирования;

Примечание – Поправка не применяется в кодировании типа, если КК неприменима к этому типу.

2) спецификацию ECN результатов команды кодирования;

3) любой другой простой и встраиваемый оператор результатов КК;

4) любую комбинацию из 1) – 3).

Примечания

1 Пока опция 3) является желаемой, может возникнуть необходимость использования опций 1) или 4), чтобы не задерживать одобрение КК и добавить другие опции позже.

2 Выбор опций 1) – 4) определяется тем, является ли спецификация достаточно точной, чтобы позволить взаимодействовать имплементациям кодеров и декодеров, которые должны быть получены.

h) утверждение, что пункты или подпункты Рек. МСЭ-Т X.691 ИСО/МЭК 8825-2, которые подлежат коррекции, являются отличными от исправленных любой ранее одобренной КК, у которой есть перекрывающийся список типов, к которым они могут быть применены.

7 Состояние предложения КК PER во время процесса одобрения

7.1 Предложение КК PER может принадлежать одной из следующих категорий:

– **Рассмотрено, но отклонено (ОТКЛОНЕНО)**: Было некоторое (задокументированное) обсуждение, но это предложение КК PER вряд ли будет развиваться в настоящее время.

– **Предложенное как возможное полезное (ВОЗМОЖНО)**: Предложение КК PER ожидает дальнейшее обсуждение.

– **В активной разработке (НЕОБХОДИМО)**: Существует соглашение, что КК PER, выполняющая данную функцию необходима, но точная спецификация все еще находится в статусе разработки. Работа направлена на доработку спецификации КК PER.

– **Условно одобрено (ГОТОВО)**: Полная спецификация доступна и готова к реализации. Данное решение одобрено соответствующей рабочей группой или соответствующей подкомиссией на пленарном совещании.

– **Одобрено (ОДОБРЕНО)**: Полная спецификация КК PER была опубликована в состоянии ГОТОВО в течение, по крайней мере, шести месяцев, и была одобрена и соответствующей рабочей группой, и соответствующей подкомиссией на пленарном совещании.

Примечание – Предложения КК PER в категориях **НЕОБХОДИМО**, **ГОТОВО** и **ОДОБРЕНО** публикуются службой регистрации (см. 9). Предложения КК PER в категориях **ОТКЛОНЕНО** и **ВОЗМОЖНО** записываются совместной группой МСЭ-Т и ИСО/МЭК СТК1 для АСН.1 в полагающихся документах.

7.2 Предложение КК PER обычно переходит из первоначального предложения в категорию либо **ОТКЛОНЕНО**, либо **ВОЗМОЖНО**, либо **НЕОБХОДИМО**, и затем в категорию **ГОТОВО** и позднее – **ОДОБРЕНО**. Однако на любой стадии до перемещения в категорию **ОДОБРЕНО** оно может быть перемещено в категорию **ОТКЛОНЕНО**.

Примечание – Обычно это происходит при выявлении проблем с его реализацией.

7.3 Как только предложение КК PER попадает в категорию **ОДОБРЕНО** (КК PER одобрено), оно должно быть изменено или удалено с помощью предложения изменения КК PER, которое должно пройти через все стадии с таким же процессом одобрения (описанным в 8) до того, как оно приведет к удалению или изменению КК PER категории **ОДОБРЕНО**.

8 Процесс одобрения

8.1 Любой участник соответствующей рабочей группы и любой участник соответствующей подкомиссии могут разрабатывать новые предложения КК PER, представляя информацию, как минимум, в соответствии с пунктом 6.1 a) – d) в формате приложения А. Дополнительный текст свободной формы с предложениями для e) – g) может быть включен и заявители нового EI призываются предоставить полную информацию, требуемую для 6.1.

8.2 Обсуждение предложения должно иметь место на следующей основной или промежуточной возможной встрече соответствующей рабочей группы или соответствующей подкомиссии. Предложение после обсуждения должно быть отнесено к категории:

a) **ОТКЛОНЕНО**: В протоколе встречи должны быть зафиксированы причины отказа;

b) **ВОЗМОЖНО**: Детали предложения КК PER должны быть перечислены в полагающемся документе совместной группы МСЭ-Т и ИСО/МЭК СТК1 для АСН.1 с записью предложенных КК PER, с датами, когда это было предложено, а также рассмотрено, информация о заявителе и все соответствующие обсуждения;

c) **НЕОБХОДИМО**: Детали предложения КК PER должны быть опубликованы (см. 9).

8.3 Предложение КК PER должно перейти в категорию **НЕОБХОДИМО** на любой промежуточной или основной встрече соответствующей рабочей группы или соответствующей подкомиссии, если присутствующие на встрече единогласны, и если присутствует текст, доступный для широкой публики с информацией, требуемой для 6.1 a) — f), одобренный на встрече. Детали КК PER должны быть опубликованы.

Примечание – Этой информации достаточно для создания синтаксических инструментов и для включения пользователями КК PER в их проект спецификации.

8.4 Предложение КК PER должно перейти в категорию **ГОТОВО** на любой основной встрече соответствующей рабочей группы или соответствующей подкомиссии, если на данной встрече принято официальное решение, основанное на доступности и анализе полного текста для всех составляющих 6.1 a) — h), что предложение КК PER одобрено этой соответствующей подкомиссией или соответствующей рабочей группой. Рассмотренные детали КК должны быть опубликованы.

8.5 Предложение КК PER должно перейти в категорию **ОДОБРЕНО**, если выполнены следующие условия:

a) предложение КК PER было в категории **ГОТОВО** в течение, по крайней мере, шести месяцев; и

b) соответствующая рабочая группа и соответствующая подкомиссия приняли официальное решение об одобрении КК PER на пленарном совещании.

9 Публикация службой регистрации

9.1 Вся регистрация и публикация должны проходить через веб-страницу на веб-сайте соответствующей рабочей группы, поддерживаемом Телекоммуникационным Бюро Стандартизации ITU, которое выступает в роли официальной службы регистрации. Новое содержание данной страницы должно обеспечиваться совместной группой МСЭ-Т и ИСО/МЭК СТК1 для АСН.1 в случае изменения статуса предложения КК PER (см. раздел 8, когда такие изменения могут произойти).

Примечания

1 Веб-страница во время публикации может быть расположена по URL адресу http://www.itu.int/ITU-T/studygroups/com17, в разделе «Registration — Assignment», под заголовком «ASN.1 PER EIs».

2 При перемещении данных страниц их новое расположение может быть получено отправкой письма по электронной почте по адресу tsbmail@itu.int с просьбой установить контакт с докладчиком ITU-T, ответственным за обслуживание ITU-T Rec. X.695 | ISO/IEC 8825-6.

9.2 Все предложения КК PER на **НЕОБХОДИМОМ**, **ГОТОВОМ** или **ОДОБРЕННОМ** этапе должны быть опубликованы.

9.3 Все предложения по изменению КК PER на **НЕОБХОДИМОМ** или **ГОТОВОМ** этапе должны быть опубликованы.

9.4 Перемещение предложение по изменению КК PER на этап **ОДОБРЕННО** должно быть отражено в архивной записи предыдущего КК PER и соответствующим изменением к измененному КК PER с объяснением любых рассмотрений обратной совместимости.

10 Ограничения на использование команд кодирования PER

10.1 Приложение КК PER может препятствовать тому, чтобы некоторые абстрактные значения типа были закодированы PER. В зависимости от предполагаемого приложения это может иметь или не иметь значения. Однако желательно дополнительно обеспечить явное ограничение на тип для гарантии того, что все дозволенные значения могут быть закодированы по всем правилам кодирования.

10.2 В случае, когда КК PER неявно запрещает кодирование некоторых абстрактных значений, спецификации КК должны указывать на то, что использование этого КК ограничивает абстрактные значения, которые могут быть закодированы.

Пример 1 – Приложение КК PER для завершения строки ASCII с конечным знаком NULL не может быть закодировано PER, если строка содержит символ NULL. Разработчик может применять нормальное ограничение для сокращения строки до символов без NULL.

Примечание – Это делает спецификацию более избыточной и, возможно, менее четкой, но гарантирует, что между различными правилами кодирования возможна передача.

Пример 2 – Приложение КК PER для использования 16-битовых полей для кодирования ЦЕЛОГО ЧИСЛА, если ЦЕЛОЕ ЧИСЛО является чрезмерно большим, не может быть закодировано с использованием правил PER, для этого необходимо рассматривать возможность ограничения диапазона, который может быть закодирован в 16-битовом поле.

10.3 КК PER не должен применяться к типу, расширенному для кодирования PER (см. Рек. МСЭ-Т. X.691 ИСО/МЭК 8825-2, 3.7.11).

Примечание – Это применимо непосредственно к типу. Расширяемость компонента не ограничивает приложение КК PER по отношению к другим компонентам или к типу.

11 Присвоение КК PER к типу ACH.1, используя префикс типа

11.1 Команды кодирования PER могут быть присвоены (или удалены из) типам ACH.1 (используя любую из альтернатив процедуры «EncodingInstruction») в префикс типа PER.

Примечание – Эффект многократных присвоений команд кодирования описывается в 13.

11.2 Процедура PER «EncodingInstruction»:

```
EncodingInstruction ::=
        PositiveInstruction
        | NegatingInstruction
PositiveInstruction ::=
        IdentifyingKeyword
        EIDetail
NegatingInstruction ::=
```

NOT PositiveInstruction

IdentifyingKeyword ::= encodingreference

11.3 «IdentifyingKeyword» имеет тот же самый синтаксис, что и у «encodingreference» (см. Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1, п. 12.25), но имеет различную семантику. Его значение должно быть определено для каждой КК PER и должно быть однозначно в пределах всего набора КК PER. Целью этого является идентификация определенной КК PER.

11.4 Процедура «EIDetail» должна быть определена для каждой КК (см. 6.1), и не должна содержать лексические элементы «[» и «]».

11.5 Команда кодирования в префиксе типа (или в секции контроля кодирования — см. 12.1.5) может быть положительной командой, использованной для добавления или замены команды кодирования (использование «PositiveInstruction»), или отрицательной командой, использованной для отмены одной или более связанных команд кодирования (использование «NegatingInstruction»).

11.6 Если «Type» в «TypeAssignment» (см. Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1, п. 16.1), имеет заключительные команды кодирования, все применения соответствующего «typereference» (в модуле, содержащем «TypeAssignment» или в каком-либо другом модуле), наследуют его заключительные команды кодирования.

12 Присвоение команд кодирования PER, используя секцию контроля кодирования

12.1 Список присвоения команд кодирования

12.1.1 Команды кодирования PER могут также быть присвоены типам ACH.1 в секции контроля кодирования PER, используя процедуру «EncodingInstructionAssignmentList».

12.1.2 Процедура PER «EncodingInstructionAssignmentList»:

EncodingInstructionAssignmentList ::=
TargettedEncodingInstruction
EncodingInstructionAssignmentList ?
TargettedEncodingInstruction ::=
"[" EncodingInstruction "]"
TargetList

12.1.3 Процедура «EncodingInstruction» описана в 11.2.

12.1.4 Каждое применение процедуры «EncodingInstruction» в секции контроля кодирования присваивает ту команду кодирования PER для появления «Type», которая идентифицирована в «TargetList» в «TargettedEncodingInstruction». Процедура «TargetList» и идентифицируемые ею цели описаны в 12.2.

12.1.5 К командам кодирования в секции контроля кодирования применяются также 11.5 и 11.6.

12.2 Идентификация целей для команд кодирования PER, используя целевой список

12.2.1 Общие правила

12.2.1.1 Все цели являются осуществлением процедуры «Type» внутри модуля ACH.1.

Примечание – Многократные цели могут быть описаны в том же самом или в различных присвоениях типа ACH.1. Цель, которая является целым модулем, может быть определена, так же как все возникновения внутри модуля встроенного типа или конструктора. Таким образом, единственный «EncodingInstruction» (использующий секцию контроля кодирования PER) может быть использован для присвоения определенной команды кодирования PER всем типам в модуле ACH.1, которые требуют присвоения данной команды кодирования.

12.2.1.2 В идентификации цели(ей) для присвоения команды кодирования PER используется процедура «TargetList», которая описана в следующих подпунктах.

12.2.1.3 Процедура «TargetList **TargetList ::=**

Targets "," +

| empty

Targets ::=

TypeIdentification

| BuiltInTypeIdentification

| IdentifiersInContext

12.2.1.4 Если «TargetList» — список одного или более «Targets» процедур, то каждая из «Targets» идентифицирует одну или более целей («Type», которым присваивается команды кодирования).

12.2.1.5 Команда кодирования PER присваивается всем типам, идентифицированным с помощью «TargetList» как описано в 12.2.1.9 — 12.2.1.14.

Примечание – Многократная идентификация в целевом списке допустима для данного «Type». В таких случаях применяется раздел 13.

12.2.1.6 (Справочно) Идентификация цели(ей) (и возможная информация о квалификации) процедуры «Targets» использует одну из четырех основных форм:

a) использование «typereference» (см. 12.2.2) возможно в сопровождении списка идентификаторов, разделенных точкой (или звездочкой, чтобы обозначить единственный компонент последовательности или набора), идентифицируя также:

1) «Type» в присвоении типа (если нет никакого разделенного точкой списка идентификаторов);

2) «Type» в компоненте определения типа (который может содержать высокоуровневые компоненты, представленные конструктором **COMPONENTS OF** – см. 12.2.1.11);

b) использование **ALL** как последнего идентификатора в форме a), идентифицируя все «Type», представленные в определении типа (который идентифицируется предыдущей ссылкой типа и разделенным точкой списком идентификаторов);

c) использование «BuiltInTypeName» (см. 12.2.3), идентифицируя все «Type» в модуле, которые описываются при помощи соответствующего встроенного имени типа или конструктора;

d) использование списка «identifier» сопровождаемых **IN** (или **ALL** сопровождаемых **IN,** или **COMPONENTS** сопровождаемых **IN**) и формы из вышестоящего пункта a) (см. также 12.2.4), идентифицируя:

1) «Type» идентифицированных компонентов формы a);

2) (использование **ALL**) все «Type», которые дословно появляются внутри «Type», идентифицированного формой a);

3) (использование **COMPONENTS**) все «Type», которые являются высокоуровневыми компонентами «Type», идентифицированного формой a).

Примечания

1 Термин «определение типа» («type definition»), используемый в a) и b), подчеркивает, что только дословно представленные идентификаторы могут использоваться. Идентификаторы не могут использоваться, если «Type» является ссылкой типа.

2 В общем случае компонент может сослаться на использование a) или d). Если существует более одного компонента типа, для которого необходима ссылка, лучше использовать d), так как он менее объемный по содержанию, но возможно использование и a). Это лишь вопрос выбранного стиля.

12.2.1.7 Тип «bitstring» (битовая строка) или «octetstring» (октет-строка) с ограничением содержимого, которое содержит тип, должны быть обработаны как тип с единственным компонентом, используя «*» как компонентный идентификатор для присвоения целевой команды к «Type» в ограничении содержимого.

12.2.1.8 Определение типа, которое является последовательностью или набором, также обрабатывается как тип с единственным компонентом, используя «*» как компонентный идентификатор для присвоения целевой команды к «Type», который является компонентом последовательности или набора.

Примечание – Возможно также идентифицировать этот единственный компонент, используя (если есть) компонентный идентификатор.

12.2.1.9 Если цель – использование фиктивного параметра параметризованного типа, цель

наследует заключительные команды кодирования фактического параметра прежде, чем закодировать предназначение команд, которым присваивается фиктивный параметр. Спецификация является допустимой только если получающиеся заключительные команды кодирования для всех инстанцирований параметризованного типа являются допустимыми.

Примечания

1 Если параметризованный тип экспортирован, заключительные команды кодирования для его фиктивных параметров переносят с ним.

2 Нет никаких механизмов, дающих возможность присвоения команд кодирования, непосредственно к «Type» из действующих параметров в инстанцировании параметризованного типа.

12.2.1.10 Если целью является «SelectionType», цель наследует заключительные команды кодирования альтернативы, выбранной для типа выбора, на который ссылается тип выбора, после которого назначаются команды кодирования, присвоенные «SelectionType».

12.2.1.11 Если цель является компонентом полученным в результате преобразования **COMPONENTS OF**, цель наследует заключительные команды кодирования компонента типа, на который ссылаются **COMPONENTS OF,** после чего команды кодирования, присвоенные произведенным компонентам, присваиваются **COMPONENTS OF**. При этом игнорируются любые команды кодирования для «Type», из которого извлекаются компоненты.

12.2.1.12 Если процедура «Targets» использует «TypeIdentification», то цели идентификации описываются в 12.2.2.

12.2.1.13 Если процедура «Targets» использует «BuiltInTypeIdentification», то цели идентификации описываются в 12.2.3.

12.2.1.14 Если процедура «Targets» использует «IdentifiersInContext», то цели идентификации описываются в 12.2.4.

Пример – Следующий фрагмент показывает определение типа ACH.1, сопровождаемое двумя различными способами присвоения команд кодирования PER в секции контроля кодирования, и наконец, то же самое определение типа ACH.1 с присвоенными командами кодирования PER, используя префиксы типа. Все три подхода приводят к одному и тому же кодированию.

Примечание – Более подробно пример рассмотрен в приложениях А и В.

Определение типа:

```
My-Type ::= SEQUENCE {
                field1 INTEGER,
                field2 CHOICE {
                        first SEQUENCE OF INTEGER,
                        second SEQUENCE OF OBJECT IDENTIFIER } }
```

Команды кодирования PER для применения EI1 и EI2 в секции контроля кодирования могут быть:

```
[EI1] field1 IN My-Type
        [EI2] first IN My-Type.field2
```

Альтернативно они могут быть:

```
[EI1] My-Type.field1
        [EI2] My-Type.field2.first
```

Определение типа с префиксами типа:

```
My-Type ::= SEQUENCE {
                field1 [EI1] INTEGER,
                field2 CHOICE {
                        first [EI2] SEQUENCE OF INTEGER,
                        second SEQUENCE OF OBJECT IDENTIFIER } }
```

12.2.2 Целевая идентификация, использующая ссылку типа АСН.1 и идентификаторы

12.2.2.1 Процедура «TypeIdentification»:

```
TypeIdentification ::=
        ALL
        | typereference ComponentReference ?

ComponentReference ::=
                "."
                ComponentIdList

ComponentIdList ::=
                ComponentId "." +

ComponentId ::=
                identifier
                | "*"
        | ALL
```

12.2.2.2 «TypeIdentification» **ALL** идентифицирует все «Type» в «TypeAssignment» в модуле.

12.2.2.3 «typereference» (см. Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1, п. 12.2) должно быть ссылкой типа, которая определяется в модуле.

12.2.2.4 Символ «*» идентифицирует «Type» (единственного) компонента последовательности или набора типа, либо тип в ограничении содержимого, который содержит «Type».

Примечание – Эта форма может использоваться, даже если последовательность или набор компонента имеет идентификатор, но использование «identifier» (см. Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1, п. 12.3) будет предпочтительней.

12.2.2.5 Если **ALL** будет использоваться в качестве «ComponentId», то это должен быть последний компонент «ComponentId» в «ComponentIdList».

12.2.2.6 Если первый компонент «ComponentId» в «ComponentIdList» является идентификатором, который присутствует (или следует из использования **COMPONENTS OF**) как компонентный идентификатор в «Type», идентифицированном «typereference», то это определяет «Type» компонента. Если это не идентификатор, который присутствует (или следует из использования **COMPONENTS OF**) как компонентный идентификатор в «Type», идентифицированном «typereference», то это появление «TypeIdentification» допустимо, но не идентифицирует никакой цели.

Примечание – Это требуется, чтобы тип, на который ссылается «typereference», был последовательностью, набором, выбором в последовательности (или наборе) определения типа или был определением типа «bitstring» или «octetstring» с ограничением содержимого, которое содержит «Type».

12.2.2.7 Если последующий «ComponentId» (кроме последнего) в «ComponentIdList» является идентификатором, который дословно присутствует как компонентный идентификатор в «Type», идентифицированном предыдущим «ComponentId», то это идентифицирует предыдущий «ComponentId», но такое появление «TypeIdentification» допустимо, но не идентифицирует никакой цели.

Примечание – Первое применение «ComponentId» может обратиться к компонентам, представленным **COMPONENTS OF**. Компоненты этих компонентов не могут быть идентифицированы последующими «ComponentId».

12.2.2.8 Последний «ComponentId» в «ComponentIdList» (если представлен):

a) идентификатор, который дословно присутствует как компонентный идентификатор в «Type», идентифицированном предыдущим «ComponentId», тогда это идентифицирует «Type» этого компонента, и команда кодирования должна быть присвоена этому «Type»;

b) ключевое слово **ALL**, в этом случае команда кодирования должна быть присвоена всем «Type», присутствующим в определении типа, идентифицированном предыдущим «ComponentId», который должен быть типом с одним или более компонентами.

12.2.3 Целевая идентификация, использующая встроенное имя типа

12.2.3.1 Процедура «BuiltInTypeIdentification» имеет вид:

```
BuiltInTypeIdentification ::=
                BIT STRING
                | BOOLEAN
                | CHARACTER STRING
                | CHOICE
                | DATE
                | DATE-TIME
                | DURATION
                | EMBEDDED PDV
                | ENUMERATED
                | EXTERNAL
                | GeneralizedTime
                | INSTANCE OF
                | INTEGER
                | NULL
                | ObjectDescriptor
                | OBJECT IDENTIFIER
                | OCTET STRING
                | REAL
                | RELATIVE-OID
                | SEQUENCE
                | SEQUENCE OF
                | SET
                | SET OF
                | TIME
                | TIME-OF-DAY
                | UTCTime
                | RestrictedCharacterStringType
```

12.2.3.2 Процедура «BuiltInTypeIdentification» определяет, что команда кодирования должна быть применена ко всем текстовым появлениям внутри модуля соответствующего встроенного типа или типа определенного, используемым соответствующим конструктором.

12.2.3.3 «RestrictedCharacterStringType» определяется в Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1, п. 41.

12.2.4 Применение идентификаторов в контексте

12.2.4.1 Процедура «IdentifiersInContext» имеет вид:

```
IdentifiersInContext ::=
                IdentifierList
                IN
                TypeIdentification

IdentifierList ::=
                identifier "," +
        | ALL
                | COMPONENTS
```

12.2.4.2 «TypeIdentification» описывается в 12.2.2 и идентифицирует тип, определенный в присвоении типа в модуле или компоненте либо субкомпоненте типа, определенного в модуле.

12.2.4.3 «Type», идентифицированный «TypeIdentification», должен быть последовательностью, набором или типом выбора и используется в целях этого пункта идентифицированным «Type».

Примечание — «TypeIdentification» в «IdentifiersInContext» не может использоваться для последовательности или набора типа. Такое использование запрещается, поскольку имеет более емкое содержание, чем прямое использование «TypeIdentification» в «Targets».

12.2.4.4 Каждый «identifier» (см. Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1, п. 12.3) в «IdentifierList»

будет «identifier» компонента идентифицированного «Type». Команда кодирования PER присваивается «Type» всех компонентов идентифицированного «Type», у которых есть компонент «identifier» в «IdentifierList».

12.2.4.5 Применение **ALL** для «IdentifierList» определяет, что все дословно существующие компоненты (и все дословно существующие компоненты этих компонентов на любом уровне) в идентифицированном «Type» являются целями, которым присваиваются команды кодирования PER.

12.2.4.6 Использование **COMPONENTS OF** для «IdentifierList» определяет, что все компоненты (на первом уровне) идентифицированного «Type» являются целями, которым присваивается команды кодирования PER.

13 Многократные присвоения команд кодирования PER

13.1 Порядок, в котором рассматривают многократные присвоения

13.1.1 У «Type», который не является «typereference», есть первоначально пустой набор связанных команд кодирования.

13.1.2 У «Type», который является «typereference» (который может быть импортирован), есть первоначальный набор заключительных команд кодирования «Type», который был ему присвоен, когда это было определено.

13.1.3 Целевые команды кодирования для «Type» (использующие секцию контроля кодирования) присваиваются в порядке, в котором целевые команды кодирования появляются в секции контроля кодирования. Если «Type» будет идентифицирован больше чем одним элементом «TargetList» (см. 12.2), то это должно быть обработано как многократные присвоения той же самой команды кодирования к тому же «Type» в порядке, в котором элементы возникают в «TargetList».

Примечание – Влияние 13.1.2 и 13.1.3 проявляется в том, что целевые присвоения к «Type» в «TypeAssignment» всегда переопределяется целевым присвоением к «Type», определенным, используя соответствующий «typereference», независимо от того, который предназначался для присвоения, окажется первым в разделе управления кодированием. Однако если целевое присвоение будет применено ко всем компонентам типа и так же к отдельному компоненту этого типа, то результат будет зависеть от порядка команд кодирования в секции контроля кодирования.

13.1.4 Снабженные префиксом команды кодирования (использующие префикс типа), присвоенные типу, рассматривают далее, команды кодирования с самым правым (внутренним) префиксом рассматривают сначала, и команды кодирования с крайним левым (наиболее удаленным) префиксом рассматривают последними.

13.1.5 Как определено в 12.2.1.9, команды кодирования присваиваются фиктивному параметру только после того как заключительные команды кодирования для действительного параметра были определены.

13.1.6 Как определено в 12.2.1.10 и 12.2.1.11, «SelectionType» и компоненты, произведенные преобразованием **COMPONENTS OF**, наследуют сначала заключительные команды кодирования исходного типа, и затем имеют команды кодирования, предназначенные для их применения.

13.1.7 Каждое присвоение команды кодирования создает новый набор связанных команд кодирования, что описано в 13.2 — 13.3.

13.2 Результат присвоения команд кодирования отрицания

Все присвоения команд кодирования отрицания приводят к удалению (от набора связанных команд кодирования) всех предыдущих команд кодирования, и набор становится пустым.

Примечание – Команды кодирования отрицания никогда не становятся частью набора связанных команд кодирования.

13.3 Многократное присвоение команд кодирования PER

Примечание – Многократное присвоение команд кодирования PER бывает редко; кроме того, команда кодирования присваивается глобально и определенным типам или компонентам. Этот подпункт описывает правила, если происходит многократное присвоение команд кодирования PER.

13.3.1 Присвоения положительных команд кодирования приводят к дополнению (к набору связанных команд кодирования) этой команды кодирования PER, если нет никой другой связанной

команды кодирования с таким же «IdentifyingKeyword».

13.3.2 Если есть команда кодирования с тем же самым «IdentifyingKeyword» в наборе связанных команд кодирования, то такая команда кодирования удаляется из набора, и присвоенная команда кодирования PER добавляется.

Примечание – Если команды кодирования присваиваются глобально в секции контроля кодирования, с намерением переопределить их в конкретных случаях, тогда переопределение должно быть сделано, используя или префикс типа, или более позднюю команду кодирования в секции контроля кодирования.

13.3.3 Если тип появляется в «ContentsConstraint» или в «TypeConstraint», то заключительные команды кодирования (как определено вышеупомянутыми правилами) используются в определении типа кодирования. Если тип появляется в каком-либо другом ограничении ACH.1, то все связи команд кодирования сбрасываются.

Приложение А
(справочное)

Пример приложения КК PER с использованием команд кодирования, снабженных префиксом

А.1 Это приложение содержит пример использования снабженных префиксом КК PER, чтобы произвести изменение кодировок PER для типов в модуле.

А.2 Пример основан на раннем тексте части стандарта ИСО/МЭК. Окончательная версия стандарта значительно отличается по определенному абстрактному синтаксису; таким образом, пример в настоящем стандарте должен использоваться исключительно в целях иллюстрации.

А.3 Для ясности команды кодирования PER (и комментарии, связанные с ними) показывают серым цветом.

А.4 Модуль, определяющий использование снабженных префиксом КК PER:

```
SignatureSignRecordFormatModule
{iso standard 19794 signature-sign(7) modules(0) record-format(0) version(0)}
DEFINITIONS
PER INSTRUCTIONS
-- This specifies that PER Encoding Instructions are to be applied
AUTOMATIC TAGS ::=
BEGIN
        SignatureSignBlock ::= SEQUENCE {
                header Header,
                body Body}
        Header ::= SEQUENCE {
                        formatId [NULL]
-- This specifies that the IA5String is to
-- be followed by a zero(NULL) octet in
-- the encoding.
                                                IA5String ("SDI"),
                        standardVersion [NULL]
                                        -- As above.
                                                IA5String (SIZE (3))
                                -- "10" (space-one-zero) for this version --,
                        channelInclusions ChannelInclusions,
                        channelDescriptions ChannelDescriptions}
ChannelInclusions ::= SEQUENCE {
                        x-included BOOLEAN,
                        y-included BOOLEAN,
                        z-included BOOLEAN,
                        vX-included BOOLEAN,
                        vY-included BOOLEAN,
                        aX-included BOOLEAN,
                        aY-included BOOLEAN,
                        t-included BOOLEAN,
                        dt-included BOOLEAN,
                        f-included BOOLEAN,
                        s-included BOOLEAN,
                        tX-included BOOLEAN,
                        tY-included BOOLEAN,
                        az-included BOOLEAN,
                        el-included BOOLEAN,
                        r-included BOOLEAN}
                                (WITH COMPONENTS
                                    {x-included (TRUE),
                                     y-included (TRUE)})
ChannelDescriptions ::= [OPTIONALITY-IN Header.channel-inclusions]
-- This specifies that the optionality bit-map is
-- taken from the channel inclusions. The
```

```
--channel-inclusions structure is needed because
-- the same bit-map controls the optionality
-- in each SamplePoint SEQUENCE in the
-- SamplePoints SEQUENCE OF. It is also
-- desirable to make the bit-map
-- application-visible
                    SEQUENCE {
                            x SignedChannelDescr OPTIONAL,
                            y SignedChannelDescr OPTIONAL,
                            z UnsignedChannelDescr OPTIONAL,
                            vX SignedChannelDescr OPTIONAL,
                            vY SignedChannelDescr OPTIONAL,
                            aX SignedChannelDescr OPTIONAL,
                            aY SignedChannelDescr OPTIONAL,
                            t UnsignedChannelDescr OPTIONAL,
                            dt UnsignedChannelDescr OPTIONAL,
                            f UnsignedChannelDescr OPTIONAL,
                            s UnsignedChannelDescr OPTIONAL,
                            tX SignedChannelDescr OPTIONAL,
                            tY SignedChannelDescr OPTIONAL,
                            az UnsignedChannelDescr OPTIONAL,
                            el UnsignedChannelDescr OPTIONAL,
                            r UnsignedChannelDescr OPTIONAL}
                              (CONSTRAINED BY {ChannelInclusions
-- Each element can be present if and only if permitted by
-- the ChannelInclusions -- })
SignedChannelDescr ::= SEQUENCE {
                            reserved INTEGER (0..8),
                            scalingValue ScalingValue OPTIONAL,
                            min SignedInt16 OPTIONAL,
                            max SignedInt16 OPTIONAL,
                            mean SignedInt16 OPTIONAL,
                            std UnsignedInt16 OPTIONAL}
                            UnsignedChannelDescr ::= SEQUENCE {
                            reserved INTEGER (0..8),
                            scalingValue ScalingValue OPTIONAL,
                            min UnsignedInt16 OPTIONAL,
                            max UnsignedInt16 OPTIONAL,
                            mean UnsignedInt16 OPTIONAL,
                            std UnsignedInt16 OPTIONAL}
UnsignedChannelDescr ::= SEQUENCE {
                            reserved INTEGER (0..8),
                            scalingValue ScalingValue OPTIONAL,
                            min UnsignedInt16 OPTIONAL,
                            max UnsignedInt16 OPTIONAL,
                            mean UnsignedInt16 OPTIONAL,
                            std UnsignedInt16 OPTIONAL
ScalingValue ::= SEQUENCE {
                            exponent [ENCODE-DIRECTLY] INTEGER (-16..15),
                                        -- This ensures a two's complement
                                        -- encoding, not an encoding from
                                        -- the base of -16.
                            fraction INTEGER (0..2047)}
Body ::= [SIZE 8] SEQUENCE {
            -- 8 bit optionality bit-map, with only one bit used. Other
            -- bits will be set to zero by encoders, ignored by decoders.
                samplePoints [LENGTH 3] [COUNT-OCTETS] SEQUENCE
                                -- Prevents optimisation for short
                                -- iterations, forcing 3 octets in
                                -- all cases
```

```
                                SIZE (0..16777215) OF SamplePoint,
            extendedData [TERMINATED-BY-CARRIER]
                                -- The end of this octet string can only
                                -- be determined by running out of the
                                -- input buffer provided by the carrier
                                -- protocol. It is a feature of this BDB
                                -- format that it relies on the CBEFF
                                -- carrier to delimit it
                                OCTET STRING OPTIONAL}
SamplePoint ::=
            [OPTIONALITY-IN Header.channelInclusions]
              -- As above
            SEQUENCE {
            x SignedInt16 OPTIONAL,
            y SignedInt16 OPTIONAL,
            z UnsignedInt16 OPTIONAL,
            vX SignedInt16 OPTIONAL,
            vY SignedInt16 OPTIONAL,
            aX SignedInt16 OPTIONAL,
            aY SignedInt16 OPTIONAL,
            t UnsignedInt16 OPTIONAL,
            dt UnsignedInt16 OPTIONAL,
            f UnsignedInt16 OPTIONAL,
            s UnsignedInt8 OPTIONAL,
            tX SignedInt16 OPTIONAL,
            tY SignedInt16 OPTIONAL,
            az UnsignedInt16 OPTIONAL,
            el UnsignedInt16 OPTIONAL,
            r UnsignedInt16 OPTIONAL}
        (CONSTRAINED BY {ChannelInclusions
                                -- Each element can be present if and only if
                                -- permitted by the channel inclusions -- })
UnsignedInt16 ::= INTEGER (0..65535)
SignedInt16 ::= [ENCODE-DIRECTLY] INTEGER (-32768..32767)
                                -- As above
UnsignedInt8 ::= INTEGER (0..255)
END
```

Приложение В
(справочное)

Пример приложения КК PER с использованием целевых команд кодирования

В.1 Это приложение содержит пример использования целевых КК PER, чтобы произвести изменение кодировок PER для типов в модуле.

В.2 Пример тот же, что и в приложении А, и производит точно то же самое кодирование. Его преимущество (по отношению к приложению А) состоит в том, что он оставляет определение абстрактного синтаксиса ненарушенным и является дополнением к основной спецификации. Он также проясняет, что другие правила кодирования (и структуры данных в ядре, выполененные компилятором) позволят сделать кодирование полного спектра абстрактных значений типа. Его недостаток — то, что это менее читаемо, чем в приложении А. Разработчики протокола могут использовать или форму, или смесь (не рекомендуется!).

В.3 Модуль, определяющий использование предназначений КК PER:

```
SignatureSignRecordFormatModule
{iso standard 19794 signature-sign(7) modules(0) record-format(0) version(0)}
DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN
    SignatureSignBlock ::= SEQUENCE {
            header Header,
            body Body}
    Header ::= SEQUENCE {
                    formatId IA5String ("SDI"),
                    standardVersion IA5String (SIZE (3))
                        -- "10" (space-one-zero) for this version --,
                    channelInclusions ChannelInclusions,
                    channelDescriptions ChannelDescriptions}
    ChannelInclusions ::= SEQUENCE {
                    x-included BOOLEAN,
                    y-included BOOLEAN,
                    z-included BOOLEAN,
                    vX-included BOOLEAN,
                    vY-included BOOLEAN,
                    aX-included BOOLEAN,
                    aY-included BOOLEAN,
                    t-included BOOLEAN,
                    dt-included BOOLEAN,
                    f-included BOOLEAN,
                    s-included BOOLEAN,
                    tX-included BOOLEAN,
                    tY-included BOOLEAN,
                    az-included BOOLEAN,
                    el-included BOOLEAN,
                    r-included BOOLEAN}
                        (WITH COMPONENTS
                            {x-included (TRUE),
                            y-included (TRUE)})
    ChannelDescriptions ::=
                SEQUENCE {
                    x       SignedChannelDescr      OPTIONAL,
                    y       SignedChannelDescr      OPTIONAL,
                    z       UnsignedChannelDescr    OPTIONAL,
                    vX      SignedChannelDescr      OPTIONAL,
                    vY      SignedChannelDescr      OPTIONAL,
                    aX      SignedChannelDescr      OPTIONAL,
                    aY      SignedChannelDescr      OPTIONAL,
                    t       UnsignedChannelDescr    OPTIONAL,
```

```
                    dt UnsignedChannelDescr          OPTIONAL,
                    f UnsignedChannelDescr           OPTIONAL,
                    s UnsignedChannelDescr           OPTIONAL,
                    tX SignedChannelDescr            OPTIONAL,
                    tY SignedChannelDescr            OPTIONAL,
                    az UnsignedChannelDescr          OPTIONAL,
                    el UnsignedChannelDescr          OPTIONAL,
                    r UnsignedChannelDescr           OPTIONAL}
                      (CONSTRAINED BY {ChannelInclusions
                  -- Each element can be present if and only if permitted by
                      -- the ChannelInclusions -- })
SignedChannelDescr ::= SEQUENCE {
                    reserved       INTEGER (0..8),
                    scalingValue   ScalingValue     OPTIONAL,
                    min            SignedInt16      OPTIONAL,
                    max            SignedInt16      OPTIONAL,
                    mean           SignedInt16      OPTIONAL,
                    std            UnsignedInt16    OPTIONAL}
UnsignedChannelDescr ::= SEQUENCE {
                    reserved          INTEGER (0..8),
                    scalingValue      ScalingValue      OPTIONAL,
                    min               UnsignedInt16     OPTIONAL,
                    max               UnsignedInt16     OPTIONAL,
                    mean              UnsignedInt16     OPTIONAL,
                    std               UnsignedInt16     OPTIONAL}
ScalingValue ::= SEQUENCE {
                  exponent INTEGER (-16..15),
                  fraction INTEGER (0..2047)}
Body ::= SEQUENCE {
                  samplePoints SEQUENCE
                                SIZE (0..16777215) OF SamplePoint,
                  extendedData OCTET STRING OPTIONAL}
SamplePoint ::= SEQUENCE {
                  x          SignedInt16          OPTIONAL,
                  y          SignedInt16          OPTIONAL,
                  z          UnsignedInt16        OPTIONAL,
                  vX         SignedInt16          OPTIONAL,
                  vY         SignedInt16          OPTIONAL,
                  aX         SignedInt16          OPTIONAL,
                  aY         SignedInt16          OPTIONAL,
                  t          UnsignedInt16        OPTIONAL,
                  dt         UnsignedInt16        OPTIONAL,
                  f          UnsignedInt16        OPTIONAL,
                  s          UnsignedInt8         OPTIONAL,
                  tX         SignedInt16          OPTIONAL,
                  tY         SignedInt16          OPTIONAL,
                  az         UnsignedInt16        OPTIONAL,
                  el         UnsignedInt16        OPTIONAL,
                  r          UnsignedInt16        OPTIONAL}
            (CONSTRAINED BY {ChannelInclusions
                  -- Each element can be present if and only if
                  -- permitted by the channel inclusions -- })
UnsignedInt16 ::= INTEGER (0..65535)
SignedInt16 ::= INTEGER (-32768..32767)
UnsignedInt8 ::= INTEGER (0..255)
ENCODING-CONTROL PER
[NULL] IA5String
[OPTIONALITY-IN Header.channel-inclusions] ChannelDescriptions, SamplePoint
[ENCODE-DIRECTLY] ScalingValue.exponent, SignedInt16
[SIZE 8] Body
```

```
[LENGTH 3] Body.samplePoints
[COUNT-OCTETS] Body.samplePoints
[TERMINATED-BY-CARRIER] Body.extendedData
END
```

Приложение С
(справочное)

Сводка нотации ACH.1

Следующие лексические элементы определяются в Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1:

encodingreference
typereference
identifier

Следующая процедура определяется в Рек. МСЭ-Т X.680 ИСО/МЭК 8824-1:

RestrictedCharacterStringType

Следующие процедуры определяются в настоящем стандарте:

```
EncodingInstruction ::=
        PositiveInstruction
        | NegatingInstruction
PositiveInstruction ::=
        IdentifyingKeyword
        EIDetail
NegatingInstruction ::=
        NOT PositiveInstruction
IdentifyingKeyword ::= encodingreference
EncodingInstructionAssignmentList ::=
        TargettedEncodingInstruction
        EncodingInstructionAssignmentList ?
TargettedEncodingInstruction ::=
    "[" EncodingInstruction "]"
    TargetList
TargetList ::=
    Targets "," +
    | empty
Targets ::=
    TypeIdentification
    | BuiltInTypeIdentification
    | IdentifiersInContext
TypeIdentification ::=
    ALL
    | typereference ComponentReference ?
ComponentReference ::=
        "."
        ComponentIdList
ComponentIdList ::=
        ComponentId "." +
ComponentId ::=
        identifier
        | "*"
    | ALL
BuiltInTypeIdentification ::=
    BIT STRING
    | BOOLEAN
    | CHARACTER STRING
    | CHOICE
    | DATE
    | DATE-TIME
    | DURATION
    | EMBEDDED PDV
    | ENUMERATED
    | EXTERNAL
    | GeneralizedTime
    | INSTANCE OF
```

```
                | INTEGER
                | NULL
                | ObjectDescriptor
                | OBJECT IDENTIFIER
                | OCTET STRING
                | OID-IRI
                | REAL
                | RELATIVE-OID
                | RELATIVE-OID-IRI
                | SEQUENCE
                | SEQUENCE OF
                | SET
                | SET OF
                | TIME
                | TIME-OF-DAY
                | UTCTime
                | RestrictedCharacterStringType
IdentifiersInContext ::=
                IdentifierList
                IN
                TypeIdentification
IdentifierList ::=
                identifier "," +
        | ALL
                                COMPONENTS
```

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам Российской Федерации

Таблица ДА

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО/МЭК 8824-1:2008	IDT	ГОСТ Р ИСО/МЭК 8824-1-2001 «Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации»
ИСО/МЭК 8824-2:2008	IDT	ГОСТ Р ИСО/МЭК 8824-2-2001 «Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 2. Спецификация информационного объекта»

Примечание – В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:

IDT — идентичный стандарт

УДК 681.3:691.39 ОКС 35.100.60 П85

Ключевые слова: обработка данных, информационный обмен, сетевое взаимодействие, взаимосвязь открытых систем, АСН.1, кодирование, быстрое информационное множество

Подписано в печать 01.09.2014. Формат 60x84 1/8.
Усл. печ. л. 2,80. Тираж 34 экз. Зак. 3204

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

ФГУП «СТАНДАРТИНФОРМ»
123995 Москва, Гранатный пер., 4.
www.gostinfo.ru info@gostinfo.ru