МЕЖГОСУДАРСТВЕННЫЙ СОВЕТ ПО СТАНДАРТИЗАЦИИ, МЕТРОЛОГИИ И СЕРТИФИКАЦИИ
(МГС)

INTERSTATE COUNCIL FOR STANDARDIZATION, METROLOGY AND CERTIFICATION
(ISC)

МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ

ГОСТ ISO/IEC 24824-1— 2013

Информационные технологии

ОБЩИЕ ПРАВИЛА ПРИМЕНЕНИЯ ASN.1

Быстрые команды

Часть 1

(ISO/IEC 24824-1:2007, IDT)

Издание официальное

Москва
Стандартинформ
2015

gost.e-ecolog.ru

Предисловие

Цели, основные принципы и основной порядок проведения работ по межгосударственной стандартизации установлены ГОСТ 1.0—92 «Межгосударственная система стандартизации. Основные положения» и ГОСТ 1.2—2009 «Межгосударственная система стандартизации. Стандарты межгосударственные, правила и рекомендации по межгосударственной стандартизации. Правила разработки, принятия, применения, обновления и отмены»

Сведения о стандарте

1 ПОДГОТОВЛЕН Федеральным государственным унитарным предприятием Государственный научно-исследовательский и конструкторско-технологический институт «ТЕСТ» (ФГУП ГосНИИ «ТЕСТ»)

2 ВНЕСЕН Федеральным агентством по техническому регулированию и метрологии (Росстандарт)

3 ПРИНЯТ Межгосударственным советом по стандартизации, метрологии и сертификации (протокол от 14 ноября 2013 г. № 44)

За принятие проголосовали:

Краткое наименование страны по МК (ИСО 3166) 004—97	Код страны по МК (ИСО 3166) 004—97	Сокращенное наименование национального органа по стандартизации
Армения	AM	Минэкономики Республики Армения
Киргизия	KG	Кыргызстандарт
Россия	RU	Росстандарт

4 Приказом Федерального агентства по техническому регулированию и метрологии от 11 июня 2014 г. № 565-ст межгосударственный стандарт ГОСТ ISO/IEC 24824-1—2013 введен в действие в качестве национального стандарта Российской Федерации с 1 сентября 2015 г.

5 Настоящий стандарт идентичен международному стандарту ISO/IEC 24824-1:2007 Information technology — Generic applications of ASN.1: Fast infoset. Part 1 (Информационные технологии. Общие правила применения ASN.1. Быстрые команды. Часть 1).

Перевод с английского языка (en).

Сведения о соответствии межгосударственных стандартов ссылочным международным стандартам приведены в дополнительном приложении ДА.

Степень соответствия — идентичная (IDT)

6 ВВЕДЕН ВПЕРВЫЕ

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

© Стандартинформ, 2015

В Российской Федерации 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 . . . 2
2.1 Идентичные рекомендации и международные стандарты . . . 2
2.2 Дополнительные ссылки . . . 3
3 Термины и определения . . . 4
3.1 Термины ASN.1 . . . 4
3.2 Термины ECN . . . 4
3.3 Термины по ISO/IEC 10646 . . . 4
3.4 Дополнительные термины . . . 4
4 Сокращения . . . 5
5 Нотация . . . 6
6 Принципы построения и использования словарных таблиц . . . 6
7 Определения типов ASN.1 . . . 7
7.1 Общие положения . . . 7
7.2 Тип Document . . . 7
7.3 Тип Element . . . 13
7.4 Тип Attribute . . . 14
7.5 Тип ProcessingInstruction . . . 15
7.6 Тип UnexpandedEntityReference . . . 15
7.7 Тип CharacterChunk . . . 16
7.8 Тип Comment . . . 16
7.9 Тип DocumentTypeDeclaration . . . 17
7.10 Тип UnparsedEntity . . . 17
7.11 Тип Notation . . . 18
7.12 Тип NamespaceAttribute . . . 18
7.13 Тип IdentifyingStringOrIndex . . . 19
7.14 Тип NonIdentifyingStringOrIndex . . . 20
7.15 Тип NameSurrogate . . . 21
7.16 Тип QualifiedNameOrIndex . . . 22
7.17 Тип EncodedCharacterString . . . 23
8 Построение и обработка документа быстрого инфо-набора . . . 25
8.1 Концептуальный порядок компонентов абстрактного значения типа Document . . . 25
8.2 Таблица алфавитов с ограниченной областью распространения . . . 26
8.3 Таблица алгоритмов кодирования . . . 26
8.4 Динамические таблицы строк . . . 26
8.5 Динамические таблицы имен и идентификаторы имен . . . 27
9 Встроенные алфавиты с ограниченной областью распространения . . . 28
9.1 «Цифровой» алфавит с ограниченной областью распространения . . . 28
9.2 Алфавит с ограниченной областью распространения «дата и время» . . . 28
10 Встроенные алгоритмы кодирования . . . 28
10.1 Общие положения . . . 28
10.2 Алгоритм кодирования «hexadecimal» . . . 28
10.3 Алгоритм кодирования «base64» . . . 29
10.4 Алгоритм кодирования «short» . . . 29
10.5 Алгоритм кодирования «int» . . . 29
10.6 Алгоритм кодирования «long» . . . 30
10.7 Алгоритм кодирования «boolean» . . . 30
10.8 Алгоритм кодирования «float» . . . 30
10.9 Алгоритм кодирования «double» . . . 31
10.10 Алгоритм кодирования «uuid» . . . 31
10.11 Алгоритм кодирования «cdata» . . . 32
11 Ограничения на поддерживаемые инфо-наборы XML и некоторые упрощения . . . 32

12 Битовое кодирование типа Document 33
Приложение А (обязательное) Модуль ASN.1 и модули ECN для документов быстрого инфо-набора 35
Приложение В (обязательное) MIME-медиатип для документов быстрого инфо-набора 61
Приложение С (справочное) Описание кодирования документа быстрого инфо-набора 63
Приложение D (справочное) Примеры кодирования инфо-наборов XML как документов быстрого инфо-набора 77
Приложение Е (справочное) Присвоение значений идентификаторов объектов 100
Приложение ДА (справочное) Сведения о соответствии межгосударственных стандартов ссылочным международным стандартам (международным документам) 101
Библиография 102

МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ

Информационные технологии

ОБЩИЕ ПРАВИЛА ПРИМЕНЕНИЯ ASN.1

Быстрые команды

Часть 1

Information technologies. Generic applications of ASN.1. Fast infoset. Part 1

Дата введения — 2015—09—01

1 Область применения

В настоящем стандарте определен тип ASN.1 (по ISO/IEC 8824-1), абстрактные значения которого представляют экземпляры инфо-набора W3C XML. Также определены двоичные кодирования для указанных значений с использованием нотации контроля кодирования ASN.1 (по ISO/IEC 8825-3).

Примечание — Данные кодирования называют документами быстрого инфо-набора.

В настоящем стандарте также определены методы:

- минимизирующие размер документов быстрого инфо-набора;
- максимизирующие скорость создания и обработки документов быстрого инфо-набора;
- позволяющие (создателю документа быстрого инфо-набора) определять дополнительные обрабатываемые данные.

Первые два метода основываются на использовании словарных таблиц. Набор словарных таблиц и характер их записей полностью определены в настоящем стандарте, но их представление в памяти компьютера не входит в область применения настоящего стандарта.

Обеспечение передачи или хранения, а также формальная нотация для отображения или определения словарных таблиц, которые должны использоваться в качестве внешних словарей, также не входят в область применения настоящего стандарта.

Третий метод заключается в предоставлении дополнительных обрабатываемых данных и URI, которые идентифицируют форму и семантику этих данных. Спецификация конкретных форм дополнительных обрабатываемых данных и их использование не входят в область применения настоящего стандарта.

URI могут быть использованы для идентификации окончательных словарей, которые, в свою очередь, могут быть использованы как часть какого-либо нового исходного словаря или составлять его целиком, однако присвоение конкретных URI конкретным окончательным словарям не входит в область применения настоящего стандарта.

В настоящем стандарте определены встроенные алфавиты с ограниченной областью распространения, описаны добавление к словарным таблицам дополнительных алфавитов с ограниченной областью распространения путем нумерации и использование этих словарных таблиц для эффективного кодирования строк символов.

В настоящем стандарте определены встроенные алгоритмы кодирования для оптимального кодирования определенных строк символов и описано добавление к словарным таблицам дополнительных алгоритмов кодирования, идентифицированных URI, но определение этих дополнительных алгоритмов и связанных с ними URI не входит в область применения настоящего стандарта.

Кроме того, в настоящем стандарте определен медиатип многоцелевых расширений интернет-почты (MIME), который идентифицирует документ быстрого инфо-набора.

Издание официальное

2 Нормативные ссылки

Для применения настоящего стандарта необходимы следующие ссылочные документы. Для датированных ссылок применяют только указанное издание ссылочного документа, для недатированных ссылок применяют последнее издание ссылочного документа (включая все его изменения).

2.1 Идентичные рекомендации и международные стандарты

- ITU-T Recommendation X.667 (2004) | ISO/IEC 9834-8:2005[1] Information technology — Open Systems Interconnection — Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components (Рекомендация МСЭ-Т X.667 (2004) | ISO/IEC 9834-8:2005 Информационные технологии. Взаимосвязь открытых систем. Процедуры для работы регистрационных органов в системе OSI. Часть 8. Создание и регистрация универсально уникальных идентификаторов и их использование в качестве компонентов идентификаторов объектов ASN.1)

- ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002[2] Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation (Рекомендация МСЭ-Т X.680 (2002) | ISO/IEC 8824-1:2002 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 1. Спецификация базовой нотации)

- ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002[3] Information technology — Abstract Syntax Notation One (ASN.1): Information object specification (Рекомендация МСЭ-Т X.681 (2002) | ISO/IEC 8824-2:2002 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 2. Спецификация информационных объектов) †

- ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002[4] Information technology — Abstract Syntax Notation One (ASN.1): Constraint specification (Рекомендация МСЭ-Т X.682 (2002) | ISO/IEC 8824-3:2002 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 3. Спецификация ограничений) †

- ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002[5] Information technology — Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications (Рекомендация МСЭ-Т X.683 (2002) | ISO/IEC 8824-4:2002 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 4. Спецификация для параметризации ASN.1) †

- ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002[6] Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER) [Рекомендация МСЭ-Т X.690 (2002) | ISO/IEC 8825-1:2002 Информационные технологии. Правила кодирования ASN.1. Часть 1. Спецификация основных (BER), канонических (CER) и различительных правил кодирования (DER)] †

- ITU-T Recommendation X.691 (2002) | ISO/IEC 8825-2:2002[7] Information technology — ASN.1 encoding rules: Specification of Packed Encoding Rules (PER) [Рекомендация МСЭ-Т X.691 (2002) | ISO/IEC 8825-2:2002 Информационные технологии. Правила кодирования ASN.1. Часть 2. Спецификация правил уплотненного кодирования (PER)] †

- ITU-T Recommendation X.692 (2002) | ISO/IEC 8825-3:2002[8] Information technology — ASN.1 encoding rules: Specification of Encoding Control Notation (ECN) [Рекомендация МСЭ-Т X.692 (2002) | ISO/IEC 8825-3:2002 Информационные технологии. Правила кодирования ASN.1. Часть 3. Спецификация нотации контроля кодирования (ECN)]

- ITU-T Recommendation X.693 (2001) | ISO/IEC 8825-4:2002[9] Information technology — ASN.1 encoding rules: XML Encoding Rules (XER) [Рекомендация МСЭ-Т X.693 (2001) | ISO/IEC 8825-4:2002 Информационные технологии. Правила кодирования ASN.1. Часть 4. Правила кодирования XML (XER)] †

---

[1] Отменен. Действует ISO/IEC 9834-8:2014.

[2] Отменен. Действует ISO/IEC 8824-1:2008.

[3] Отменен. Действует ISO/IEC 8824-2:2008.

[4] Отменен. Действует ISO/IEC 8824-3:2008.

[5] Отменен. Действует ISO/IEC 8824-4:2008.

[6] Отменен. Действует ISO/IEC 8825-1:2008.

[7] Отменен. Действует ISO/IEC 8825-2:2008.

[8] Отменен. Действует ISO/IEC 8825-3:2008.

[9] Отменен. Действует ISO/IEC 8825-4:2008.

Примечание — Приведен полный перечень рекомендаций и международных стандартов по ASN.1, так как все они могут быть применены в конкретных случаях использования настоящего стандарта. Если в тексте настоящего стандарта нет прямых ссылок на какой-либо документ, то в приведенном списке данный документ помечен символом †.

2.2 Дополнительные ссылки

- ISO 8601:2004, Data elements and interchange formats — Information interchange — Representation of dates and times (ISO 8601:2004 Элементы данных и форматы для обмена информацией. Обмен информацией. Представление дат и времени)

- ISO/IEC 10646:2003[1] Information technology — Universal Multiple-Octet Coded Character Set (UCS) [ISO/IEC 10646:2003 Информационные технологии. Универсальный многооктетный набор кодированных символов (UCS)]

- The Unicode Standard, Version 4.0, The Unicode Consortium (Reading, MA, Addison-Wesley) (Стандарт Юникод, версия 4.0, Консорциум Юникода)

Примечание 1 — Графические символы (и их кодирования), определенные в стандарте Юникод, идентичны тем, которые определены в ISO/IEC 10646-1, а стандарт Юникод включен в перечень потому, что в нем определены имена управляющих символов и аббревиатура UTF-16BE.

- W3C XML 1.0:2004, Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), gost.e-ecolog.ru

gost.e-ecolog.ru XML 1.0:2004. Расширяемый язык разметки (XML) 1.0 (третья редакция). Рекомендация консорциума W3C, © [4.02.2004] Консорциум Всемирной паутины (Массачусетский технологический институт, Национальный институт исследований в области компьютерной обработки данных и автоматики, Университет Кэйо), http://www.w3.org/TR/2000/REC-xml-20040204/)

- W3C XML 1.1:2004, Extensible Markup Language (XML) 1.1, W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), gost.e-ecolog.ru

gost.e-ecolog.ru 3C XML 1.1:2004. Расширяемый язык разметки (XML) 1.1. Рекомендация консорциума W3C, © [4.02.2004] Консорциум Всемирной паутины (Массачусетский технологический институт, Национальный институт исследований в области компьютерной обработки данных и автоматики, Университет Кэйо), http://www.w3.org/TR/2000/REC-xml11-20040204/)

Примечание 2 — В перечень включены ссылки как на W3C XML 1.0, так и на W3C XML 1.1, поскольку ни один из них не является подмножеством другого. Данные ссылки использованы только в 3.4.10.

- W3C XML Information Set:2004, XML Information Set (Second Edition), W3C Recommendation, Copyright © [04 February 2004] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), gost.e-ecolog.ru

gost.e-ecolog.ru (Информационный набор XML:2004. Информационный набор XML (вторая редакция). Рекомендация консорциума W3C, © [04.02.2004] Консорциум Всемирной паутины (Массачусетский технологический институт, Национальный институт исследований в области компьютерной обработки данных и автоматики, Университет Кэйо), http://www.w3.org/TR/2004/REC-xml-infoset-20040204/)

- W3C XML Namespaces 1.0:1999, Namespaces in XML, W3C Recommendation, Copyright © [14 January 1999] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://gost.e-ecolog.ru

(Пространства имен XML 1.0:1999. Пространства имен в XML. Рекомендация консорциума W3C, © [14.01.1999] Консорциум Всемирной паутины (Массачусетский технологический институт, Национальный институт исследований в области компьютерной обработки данных и автоматики, Университет Кэйо), http://www.w3.org/TR/1999/REC-xm-lnames-19990114/)

- W3C XML Namespaces 1.1:2004, Namespaces in XML 1.1, W3C Recommendation, Copyright © [4 February 2004] World Wide Web Consortium (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), gost.e-ecolog.ru

gost.e-ecolog.ru Пространства имен XML 1.1:2004. Пространства имен в XML 1.1. Рекомендация консорциума W3C, © [4.02.2004] Консорциум Всемирной паутины (Массачусетский технологический

[1] Отменен. Действует ISO/IEC 10646:2014.

институт, Национальный институт исследований в области компьютерной обработки данных и автоматики, Университет Кэйо), http://www.w3.org/TR/2004/REC-xm-l-names11-20040204/)

Примечание 3 — В перечень включены ссылки как на пространства имен W3C XML 1.0, так и на пространства имен W3C XML 1.1, поскольку ни одно из них не является подмножеством другого. Данные ссылки использованы только в 3.4.10.

- IETF RFC 2045 (1996), Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies (IETF RFC 2045 (1996). Многоцелевые расширения интернет-почты (MIME). Часть 1. Формат текста интернет-сообщения)
- IETF RFC 2396 (1998), Uniform Resource Identifiers (URI): Generic Syntax (IETF RFC 2396 (1998). Универсальный код ресурса (URI): Основной синтаксис)
- IEEE 754-1985[1] IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754-1985. Стандарт формата представления чисел с плавающей точкой)

3 Термины и определения

В настоящем стандарте применены следующие термины по международным стандартам:

3.1 Термины ASN.1

В настоящем стандарте применены следующие термины по ISO/IEC 8824-1:
a) тип choice;
b) тип sequence;
c) тип sequence-of.

3.2 Термины ECN

В настоящем стандарте применены следующие термины по ISO/IEC 8825-3:
a) модуль определения кодирования (EDM);
b) модуль связи кодирования (ELM).

3.3 Термины по ISO/IEC 10646

В настоящем стандарте применен следующий термин по ISO/IEC 10646:
a) основная многоязычная плоскость.

3.4 Дополнительные термины

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.4.1 **Base64**: Метод кодирования, представляющий значение строки октетов в виде строки символов, в которой использован алфавит с ограниченной областью распространения из 65 символов (см. 10.3 и IETF RFC 2045).

3.4.2 **строка символов** (character string): Строка абстрактных символов по ISO/IEC 10646, не подразумевающая какого-либо способа кодирования.

3.4.3 **алгоритм кодирования** (encoding algorithm): Точная спецификация того, как эффективно закодировать в виде октетов строку символов с заданными характеристиками.

Примечание — Примером алгоритма кодирования является кодирование строки вида «-32176» в виде двоичного целого дополнения в двух октетах. Двухоктетное кодирование следует сопроводить индексом словарной таблицы, идентифицирующим этот алгоритм кодирования.

3.4.4 **внешний словарь** (external vocabulary): Набор словарных таблиц, указанный URI (см. 7.2.14).

3.4.5 **документ быстрого инфо-набора** (fast infoset document): Инфо-набор XML, представленный в соответствии с требованиями настоящего стандарта.

3.4.6 **окончательный словарь** (final vocabulary): Содержимое словарных таблиц по окончании создания или обработки документа быстрого инфо-набора.

3.4.7 **информационный элемент** (information item): Каждый из типов элементов, составляющих инфо-набор XML.

[1] Отменен. Действует IEEE 754-2008.

3.4.8 **исходный словарь** (initial vocabulary): Набор словарных таблиц, установленный информацией о заголовке документа быстрого инфо-набора, который (опционально) ссылается на внешний словарь и (опционально) предоставляет дополнительные записи таблицы.

3.4.9 **идентификатор имени** (name surrogate): Набор из трех индексов словарной таблицы (первые два являются опциональными), который используют для представления квалифицированного имени (см. 3.4.11).

3.4.10 **корректно сформированный в отношении пространств имен документ XML** (namespace-well-formed XML document): Документ W3C XML 1.0, сформированный корректно в соответствии с пространством имен W3C XML 1.0, либо документ W3C XML 1.1, сформированный корректно в соответствии с пространством имен W3C XML 1.1.

3.4.11 **квалифицированное имя** (qualified name): Набор, состоящий из свойств **[prefix]**, **[namespace name]** и **[local name]** информационного элемента **element** или информационного элемента **attribute**.

3.4.12 **алфавит с ограниченной областью распространения** (restricted alphabet): Упорядоченный набор отдельных символов по ISO/IEC 10646, который позволяет компактно кодировать любую строку символов, полностью состоящую из символов данного набора.

3.4.13 **индекс словарной таблицы** (vocabulary table index): Положительное целое значение, идентифицирующее запись в словарной таблице.

3.4.14 **словарные таблицы** (vocabulary tables): Набор концептуальных таблиц (обычно, но необязательно, создаваемых динамически), связанный с документом быстрого инфо-набора, который содержит строки символов или другую информацию и поддерживает использование обычно малых положительных целых значений (индексов словарных таблиц) для идентификации записей в таблицах.

Примечание — Примерами словарных таблиц являются таблицы, содержащие строки символов, которые являются свойством **[local name]** информационных элементов **attribute** или **element**, или строки символов, соответствующие последовательностям информационных элементов **character**, которые являются членами свойства **[children]** информационных элементов **element**.

3.4.15 **декларация XML** (XML declaration): Кодирование UTF-8 заданной строки символов (см. 12.3), которое может быть включено в начало документа быстрого инфо-набора для идентификации данного кодирования как документа быстрого инфо-набора и для того, чтобы отличить его от документов W3C XML 1.0 и документов W3C XML 1.1.

3.4.16 **инфо-набор XML** (XML infoset): Абстрактное множество данных, описывающее информацию в корректно сформированном в отношении пространств имен документе XML, как определено в инфо-наборе W3C XML.

3.4.17 **пробел XML** (XML whitespace): Один или несколько следующих символов Unicode: HORIZONTAL TABULATION (9), LINE FEED (10), CARRIAGE RETURN (13) или SPACE (32).

Примечание — Эти символы соответствуют символам «S» (пробелы) как в W3C XML 1.0, так и в W3C XML 1.1 (см. W3C XML 1.0, п. 2.3 и W3C XML 1.1, п. 2.3). Символы NEXT LINE (133) и LINE SEPARATOR (8232), которые могут встретиться в корректно сформированных в отношении пространств имен документах W3C XML 1.1 (см. W3C XML 1.1, п. 2.11), преобразуются в символы LINE FEED при обработке конца строки (см. W3C XML 1.1, п. 2.11). Когда эти символы встречаются в инфо-наборе XML, созданном из корректно сформированного в отношении пространств имен документа W3C XML 1.1, они не являются пробелами XML.

4 Сокращения

В настоящем стандарте применены следующие сокращения:

ASN.1 — Abstract Syntax Notation One (абстрактная синтаксическая нотация версии 1);

ECN — Encoding Control Notation (нотация контроля кодирования);

MIME — Multipurpose Internet Mail Extensions (многоцелевые расширения интернет-почты);

UBL — Universal Business Language (универсальный бизнес-язык);

URI — Uniform Resource Identifier (унифицированный идентификатор ресурса);

UTF-8 — Universal Transformation Function 8-bit (универсальная функция преобразования, 8 бит, по ISO/IEC 10646, Приложение D);

UTF-16BE — Universal Transformation Function 16-bit Big Endian (универсальная функция преобразования, 16 бит, по Unicode, п. 2.6);

UUID — Universally Unique Identifier (универсальный уникальный идентификатор);

XML — eXtensible Markup Language (расширяемый язык разметки).

5 Нотация

5.1 Для формального определения типов данных, кодирования которых являются документами быстрого инфо-набора, в настоящем стандарте используют нотацию ASN.1 по ISO/IEC 8824-1.

Примечание — В разделе 12 определено применение ISO/IEC 8825-3 к определению типов ASN.1, обеспечивающее кодирование документа быстрого инфо-набора на уровне битов.

5.2 В настоящем стандарте **`полужирный шрифт Courier New`** используется для нотации ASN.1, а **полужирный шрифт Arial** — для синтаксиса W3C XML и имен информационных элементов инфо-набора XML.

5.3 Имена свойств информационных элементов пишут **полужирным шрифтом Arial** и заключают в квадратные скобки (например, **[children]**).

5.4 Имена категорий строк символов (см. 8.4.2) и категорий уточненных имен пишут ПРОПИСНЫМИ БУКВАМИ.

5.5 В настоящем стандарте позиции битов внутри октета специфицируют с использованием терминов «первый бит», «второй бит» и т. д. до «восьмого бита», где первый бит является самым старшим битом октета, а восьмой бит — самым младшим.

6 Принципы построения и использования словарных таблиц

6.1 Словарные таблицы представляют собой концептуальные таблицы, отображающие индекс словарной таблицы в ее запись.

Примечание — В настоящем стандарте не определяется представление словарных таблиц в памяти компьютера и средства, с помощью которых реализация отображает индекс словарной таблицы в ее запись.

6.2 Содержимое словарных таблиц определяет создатель документа быстрого инфо-набора из инфо-набора XML.

6.3 В самом общем случае заголовок документа быстрого инфо-набора может ссылаться на набор словарных таблиц (внешний словарь) с последующей спецификацией дополнений к этим словарным таблицам с целью формирования исходного словаря для данного документа быстрого инфо-набора. В ходе создания и обработки документа быстрого инфо-набора происходят последующие добавления к словарным таблицам, в результате чего они увеличиваются, образуя окончательные словарные таблицы для данного документа.

6.4 В ходе создания и обработки документа быстрого инфо-набора некоторые словарные таблицы увеличиваются от исходного словаря до окончательного, и поэтому имена таких словарных таблиц содержат слово «динамическая». Методы удаления записей из каких-либо таблиц отсутствуют.

6.5 Индексы словарных таблиц присваивают неявно. Первая запись в любой словарной таблице имеет индекс словарной таблицы, равный единице, а каждая последующая запись в данной таблице имеет в качестве значения индекса словарной таблицы следующее по порядку целое значение. Когда в настоящем стандарте специфицировано, что в словарную таблицу должно быть что-либо добавлено, то подразумевается, что должен быть присвоен следующий доступный индекс словарной таблицы.

Примечание — Значения индексов словарных таблиц начинаются с единицы, а не с нуля потому, что значение ноль (когда оно допустимо) имеет специальный смысл «пустая строка символов» в поле, которое в противном случае может содержать индекс словарной таблицы.

6.6 Для поддержки такого неявного присвоения индексов словарных таблиц определен концептуальный порядок обработки компонентов (на любую глубину) документа быстрого инфо-набора (см. 8.1).

Примечание — Указанный порядок совпадает с порядком кодирования компонентов в документе быстрого инфо-набора. Это не обязательно подразумевает, что семантика документа обрабатывается в том же порядке. Данный порядок определен только для того, чтобы создатель и обработчик документа быстрого инфо-набора присваивали любой заданной записи словарной таблицы один и тот же индекс словарной таблицы.

6.7 Словарные таблицы применяют для разных целей (см. раздел 8), их основной функцией является обеспечение использования индексов словарных таблиц вместо записей словарных таблиц, когда эти индексы меньше по размеру (и, возможно, будут быстрее обработаны), чем записи. Некоторые встроенные записи для некоторых словарных таблиц определены в разделе 9. Эти записи всегда неявно присутствуют в указанных словарных таблицах и имеют значения индексов словарных таблиц, определенные в разделе 9.

6.8 Для некоторых категорий строк символов создатель документа быстрого инфо-набора имеет возможность выбора: добавлять или нет строку в словарную таблицу в зависимости от предполагаемого (или известного) числа появлений данной строки символов в инфо-наборе XML.

6.9 Точная форма и смысл записей словарных таблиц определены в разделе 8; в большинстве случаев записи представляют собой строки символов переменной длины, часто короткие, но потенциально длиной до 2^{32} октет.

6.10 Соответствующий требованиям создатель документа быстрого инфо-набора должен осуществлять все добавления в словарные таблицы так, как определено в 7.13.7, 7.14.6, 7.14.7 и 7.16.7. Тем самым гарантируется, что число записей в каждой словарной таблице никогда не превысит 2^{20}.

Примечание — Запись словарной таблицы может быть идентична одной или нескольким другим записям словарной таблицы. Это позволяет эффективно создавать документы быстрого инфо-набора. Однако дубликаты записей снижают эффективность передачи. Дубликаты записей не влияют на обработчика документа быстрого инфо-набора.

6.11 Соответствующий требованиям обработчик документа быстрого инфо-набора должен осуществлять все добавления в словарные таблицы так, как определено в 7.13.8, 7.14.11[1] и 7.16.8. Тем самым гарантируется, что указанные в 6.10 ограничения не будут нарушены.

7 Определения типов ASN.1

7.1 Общие положения

7.1.1 В настоящем стандарте определен набор типов ASN.1, поддерживающих представление инфо-набора XML. Корневым типом данного набора является тип **Document**.

7.1.2 Для содержимого инфо-наборов XML установлены некоторые ограничения и сделаны некоторые упрощения в представлении (см. раздел 11) для того, чтобы повысить удобство использования данной спецификации и эффективность создаваемых с ее помощью кодирований.

Примечание — Инфо-набор XML, не соответствующий установленным ограничениям, не может быть представлен в виде документа быстрого инфо-набора или корректно сформированного в отношении пространств имен документа XML.

7.1.3 В настоящем стандарте для каждого вида информационных элементов, специфицированного в инфо-наборе W3C XML, приведено определение соответствующего типа ASN.1. Это определение типа всегда является последовательностью с компонентами, соответствующими свойствам информационного элемента.

7.1.4 Некоторые свойства информационных элементов не включены в определения типов ASN.1 (см. 11.4).

7.1.5 В некоторых случаях значение свойства, не включенного в определения типов ASN.1, может быть выведено из значений других свойств того же самого информационного элемента или других информационных элементов, включенных в определения типов ASN.1. В таких случаях опущение этого свойства упрощает представление без потерь информации. Однако существуют случаи, когда значение невключенного свойства не может быть выведено из значений других свойств. Во всех подобных случаях опущение такого свойства является упрощением, не ограничивающим полезность данной спецификации для большинства случаев ее практического применения.

7.1.6 В разделе 12 определено кодирование типа **Document**.

7.2 Тип Document

7.2.1 Тип **Document** определен следующим образом:

```
Document ::= SEQUENCE {
  additional-data            SEQUENCE (SIZE(1..one-meg)) OF
    additional-datum SEQUENCE {
      id                              URI,
      data                    NonEmptyOctetString } OPTIONAL,
  initial-vocabulary         SEQUENCE {
```

[1] Ссылка в ISO/IEC 24824-1:2007 дана ошибочно, пункт 7.14.11 в оригинале отсутствует.

```
    external-vocabulary         URI OPTIONAL,
    restricted-alphabets        SEQUENCE (SIZE(1..256)) OF
      NonEmptyOctetString OPTIONAL,
    encoding-algorithms         SEQUENCE (SIZE(1..256)) OF
      NonEmptyOctetString OPTIONAL,
    prefixes                    SEQUENCE (SIZE(1..one-meg)) OF
      NonEmptyOctetString OPTIONAL,
    namespace-names             SEQUENCE (SIZE(1..one-meg)) OF
      NonEmptyOctetString OPTIONAL,
    local-names                 SEQUENCE (SIZE(1..one-meg)) OF
      NonEmptyOctetString OPTIONAL,
    other-ncnames               SEQUENCE (SIZE(1..one-meg)) OF
      NonEmptyOctetString OPTIONAL,
    other-uris                        SEQUENCE (SIZE(1..one-meg)) OF
      NonEmptyOctetString OPTIONAL,
    attribute-values            SEQUENCE (SIZE(1..one-meg)) OF
      EncodedCharacterString OPTIONAL,
    content-character-chunks    SEQUENCE (SIZE(1..one-meg)) OF
      EncodedCharacterString OPTIONAL,
    other-strings               SEQUENCE (SIZE(1..one-meg)) OF
      EncodedCharacterString OPTIONAL,
    element-name-surrogates     SEQUENCE (SIZE(1..one-meg)) OF
      NameSurrogate OPTIONAL,
    attribute-name-surrogates   SEQUENCE (SIZE(1..one-meg)) OF
      NameSurrogate OPTIONAL }
      (CONSTRAINED BY {
        -- Если присутствует компонент initial-vocabulary, то
        -- должен присутствовать как минимум один из его
        -- компонентов -- }) OPTIONAL,
  notations                   SEQUENCE (SIZE(1..MAX)) OF
                                Notation OPTIONAL,
  unparsed-entities           SEQUENCE (SIZE(1..MAX)) OF
                                UnparsedEntity OPTIONAL,
  character-encoding-scheme NonEmptyOctetString OPTIONAL,
  standalone                  BOOLEAN OPTIONAL,
  version                           NonIdentifyingStringOrIndex OPTIONAL
                                      -- Категория OTHER STRING --,
  children                    SEQUENCE (SIZE(0..MAX)) OF
    CHOICE {
      element                       Element,
      processing-instruction        ProcessingInstruction,
      comment                       Comment,
      document-type-declaration     DocumentTypeDeclaration }}
```

где значением **one-meg** является:

```
one-meg INTEGER ::= 1048576  -- Два в степени 20
```

Тип **NonEmptyOctetString**:

```
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
```

где значением **four-gig** является:

```
four-gig INTEGER ::= 4294967296 -- Два в степени 32
```

Тип **URI**:

```
URI ::= NonEmptyOctetString
```

7.2.2 Типы **EncodedCharacterString**, **NameSurrogate**, **Notation**, **UnparsedEntity**, **NonIdentifyingStringOrIndex**, **Element**, **ProcessingInstruction**, **Comment** и **Document-TypeDeclaration** определены в 7.17, 7.15, 7.11, 7.10, 7.14, 7.3, 7.5, 7.8 и 7.9 соответственно.

7.2.3 Тип **URI** должен быть URI по IETF RFC 2396.

7.2.4 Компонент **restricted-alphabets** в **initial-vocabulary** (при его наличии) должен содержать одну или несколько строк символов, каждая их которых, в свою очередь, содержит символы алфавита с ограниченной областью распространения. Каждая строка символов должна содержать как минимум два символа, и все символы в строке символов должны быть различными.

Примечание — Использование алфавита с ограниченной областью распространения для оптимизации кодирования строк символов определено в 7.17.6.

7.2.5 Компонент **encoding-algorithms** в **initial-vocabulary** (при его наличии) должен содержать один или несколько URI, каждый из которых идентифицирует алгоритм кодирования.

Примечание — В настоящем стандарте определены встроенные алгоритмы кодирования (см. раздел 10) с определенными индексами словарных таблиц; определение дополнительных алгоритмов кодирования, связанных с ними индексов словарных таблиц и способов определения таких алгоритмов не входит в область применения настоящего стандарта. Информация, необходимая для определения алгоритма кодирования, специфицирована в 8.3.3.

7.2.6 Тип **Document** представляет информационный элемент **document** инфо-набора XML. Так как все остальные информационные элементы в инфо-наборе XML являются свойствами либо этого информационного элемента, либо элементов, являющихся дочерними элементами или потомками (произвольной глубины) данного, то каждый информационный элемент **Document** полностью представляет инфо-набор XML.

Примечание — Каждый информационный элемент **Document** без ссылки на внешний словарь (см. 7.2.13) также определяет окончательный словарь, который может быть использован как внешний словарь какого-либо другого документа быстрого инфо-набора.

7.2.7 Компонент **additional-data** (при его наличии) должен содержать один или несколько компонентов **additional-datum** для обеспечения возможности дополнительных методов обработки документа быстрого инфо-набора.

Примечания

1 Примером являются данные, которые предоставляют обработчику документа быстрого инфо-набора доступ к частям документа быстрого инфо-набора без необходимости обрабатывать весь документ. Форма таких данных не стандартизирована.

2 Количество компонентов **additional-datum** ограничено 2^{20} компонентами (см. 7.2.1).

7.2.8 Каждый компонент **additional-datum** должен состоять из:

a) компонента **id** (значение типа **URI**); URI должен ссылаться на спецификацию, определяющую форму и семантику компонента **data**.

Примечание — Форма **additional-datum** может быть задана как абстрактный тип в сочетании с правилом кодирования или иным пригодным способом;

b) компонента **data**, являющегося строкой октетов, которая содержит дополнительные данные обработки.

7.2.9 Использование компонента **additional-data** подчиняется следующим условиям:

a) компонент **additional-datum** может быть проигнорирован обработчиком документа быстрого инфо-набора, если URI не распознан или дополнительная обработка рассматривается как не относящаяся к деятельности обработчика документа быстрого инфо-набора;

b) обработчик документа быстрого инфо-набора, игнорирующий все компоненты **additional-datum**, тем не менее способен создать инфо-набор XML, эквивалентный инфо-набору XML, использованному для создания документа быстрого инфо-набора.

7.2.10 Могут присутствовать несколько компонентов **additional-datum** с одним и тем же URI, и они должны быть обработаны в соответствии со спецификацией, связанной с URI.

7.2.11 Компонент **initial-vocabulary** предоставляет данные, которые (совместно с некоторыми встроенными записями таблиц) полностью определяют первоначальное содержимое таблицы алфавитов с ограниченной областью распространения (см. 8.2), таблицы алгоритмов кодирования (см. 8.3), динамических таблиц строк (см. 8.4) и динамических таблиц имен (см. 8.5) данного документа быстрого инфо-набора (исходный словарь документа быстрого инфо-набора). Исходный словарь состоит из следующих данных:

a) упорядоченного набора алфавитов с ограниченной областью распространения (см. 8.2.2), содержащего как минимум встроенные алфавиты с ограниченной областью распространения (см. раздел 9);

b) упорядоченного набора алгоритмов кодирования (см. 8.3.2), содержащего как минимум встроенные алгоритмы кодирования (см. раздел 10);

c) восьми независимых упорядоченных наборов строк символов, соответствующих восьми категориям строк символов, определенным в настоящем стандарте (см. 8.4.2); каждый из наборов содержит ноль или более строк символов одной категории;

d) двух независимых упорядоченных наборов идентификаторов имен (см. 8.5.2), соответствующих двум категориям квалифицированных имен, определенным в настоящем стандарте (см. 8.5.4); каждый из наборов содержит ноль или более идентификаторов имен одной категории.

Примечание — Исходный словарь не может быть полностью пустым, так как он всегда содержит (как минимум) встроенные алфавиты с ограниченной областью распространения и встроенные алгоритмы кодирования. Однако для документов быстрого инфо-набора не является необычной ситуация, когда исходный словарь содержит только эти данные, так как решение (создателя документа быстрого инфо-набора) о том, как использовать компонент **initial-vocabulary**, зависит от реализации, и в некоторых реализациях может быть выбрано динамическое добавление записей во все словарные таблицы (внутри тела документа быстрого инфо-набора).

7.2.12 Исходный словарь документа быстрого инфо-набора должен быть определен следующим образом:

a) если компонент **initial-vocabulary** отсутствует, то исходный словарь должен состоять только из встроенных записей таблиц, определенных в 7.2.21, 7.2.22 и в разделах 9 и 10;

b) если компонент **initial-vocabulary** присутствует, а компонент **external-vocabulary** отсутствует, то исходный словарь должен состоять из встроенных записей таблиц, определенных в 7.2.21, 7.2.22 и в разделах 9 и 10, с дополнительными записями таблиц (при их наличии), определенными в 7.2.16;

c) если компоненты **initial-vocabulary** и **external-vocabulary** присутствуют, то исходный словарь должен состоять из окончательного словаря, идентифицированного компонентом **external-vocabulary**, как определено в 7.2.13 и 7.2.14, с дополнительными записями таблиц (при их наличии), определенными в 7.2.16.

7.2.13 Компонент **external-vocabulary** идентифицирует окончательный словарь, используя один из определенных в 7.2.14 способов. Тип **URI** (см. 7.2.1) определяет окончательный словарь, который будет использован как внешний словарь одним из трех способов (см. 7.2.14).

Примечание — В настоящем стандарте не определены никакие внешние словари и URI, которые ссылаются на внешние словари. Такие внешние словари и URI могут быть определены как уполномоченные, имеющие возможность выделять URI, и могут быть согласованы в частном порядке или стандартизированы.

7.2.14 Внешний словарь может быть специфицирован одним из трех способов:

a) как окончательный словарь документа быстрого инфо-набора, который не должен сам ссылаться на внешний словарь.

Примечания

1 Хранится локально окончательный словарь или только документ быстрого инфо-набора, а окончательный словарь создается динамически в ходе обработки — зависит от конкретной реализации.

2 Ограничение, состоящее в том, что окончательный словарь документа быстрого инфо-набора со ссылкой на внешний словарь не может быть использован в качестве внешнего словаря, установлено для того, чтобы упросить реализацию и избежать циклических ссылок;

b) как корректно сформированный в отношении пространств имен документ XML, который концептуально обрабатывается следующим образом:

1) должен быть определен инфо-набор XML данного корректно сформированного в отношении пространств имен документа XML;

2) должен быть создан документ быстрого инфо-набора для этого инфо-набора XML, как определено в настоящем стандарте, но он не должен содержать компонент **initial-vocabulary**, компонент **add-to-table** типа **NonIdentifyingStringOrIndex** (см. 7.14) всегда должен быть равен **TRUE**, и ни в одной из таблиц строк не должны присутствовать кратные идентичные строки символов;

3) окончательный словарь этого документа быстрого инфо-набора становится внешним словарем.

Примечание — Хранится локально окончательный словарь или только документ быстрого инфо-набора, а окончательный словарь создается динамически в ходе обработки — зависит от конкретной реализации;

c) как набор словарных таблиц, специфицированных с использованием какого-либо другого, достаточно точного способа или текста, который должен включать в себя встроенные записи таблиц разделов 9 и 10 (с индексами словарных таблиц, определенными в указанных разделах).

Примечания

1 Спецификация нотации для определения словарных таблиц не входит в область применения настоящего стандарта.

2 Требование включать встроенные записи словарных таблиц при использовании этого способа гарантирует, что все словарные таблицы содержат встроенные записи таблиц.

7.2.15 Для внешнего словаря, определенного в соответствии с 7.2.14, всем записям таблиц, содержащим строки и имена, за исключением таблиц PREFIX и NAMESPACE NAME, должны быть присвоены последовательные индексы, начиная с 1. Записям таблиц PREFIX и NAMESPACE NAME должны быть присвоены последовательные индексы, начиная с 2. Всем алфавитам с ограниченной областью распространения, за исключением встроенных, должны быть присвоены последовательные индексы, начиная с 16. Всем алгоритмам кодирования, за исключением встроенных, должны быть присвоены последовательные индексы, начиная с 32.

7.2.16 Каждый компонент типа **NonEmptyOctetString**, **EncodedCharacterString** или **NameSurrogate** (при его наличии), который присутствует в любом из оставшихся компонентов **initial-vocabulary**, должен быть добавлен по порядку (см. 8.1) в словарную таблицу.

Таблица 1 — Соответствие идентификаторов компонентов словарным таблицам

Идентификатор компонента	Тип ASN.1 записи	Словарная таблица (см. раздел 8)
restricted-alphabets	**NonEmptyOctetString**	Таблица алфавитов с ограниченной областью распространения (см. 8.2)
encoding-algorithms	**NonEmptyOctetString**	Таблица алгоритмов кодирования (см. 8.3)
prefixes	**NonEmptyOctetString**	Таблица PREFIX (см. 8.4)
namespace-names	**NonEmptyOctetString**	Таблица NAMESPACE NAME (см. 8.4)
local-names	**NonEmptyOctetString**	Таблица LOCAL NAME (см. 8.4)
other-ncnames	**NonEmptyOctetString**	Таблица OTHER NCNAME (см. 8.4)
other-uris	**NonEmptyOctetString**	Таблица OTHER URI (см. 8.4)
attribute-values	**EncodedCharacterString**	Таблица ATTRIBUTE VALUE (см. 8.4)
content-character-chunks	**EncodedCharacterString**	Таблица CONTENT CHARACTER CHUNK (см. 8.4)
other-strings	**EncodedCharacterString**	Таблица OTHER STRING (см. 8.4)
element-name-surrogates	**NameSurrogate**	Таблица ELEMENT NAME (см. 8.5)
attribute-name-surrogates	**NameSurrogate**	Таблица ATTRIBUTE NAME (см. 8.5)

7.2.17 Значение типа **NonEmptyOctetString** должно содержать кодирование UTF-8 (см. ISO/IEC 10646, Приложение D) строки символов.

7.2.18 Таблицы алфавитов с ограниченной областью распространения и алгоритмов кодирования в исходном словаре должны содержать не более 256 записей. Все остальные таблицы должны содержать не более 2^{20} записей.

Примечание — Ограничение на количество записей должно гарантировать общие верхние границы индексов таблиц. Данное ограничение действует и тогда, когда записи таблицы добавляют динамически (см. 7.13.7, 7.14.6, 7.14.7 и 7.16.7). Данные ограничения не препятствуют кодированию любого инфо-набора XML как документа быстрого инфо-набора.

7.2.19 Встроенные алфавиты с ограниченной областью распространения имеют индексы словарных таблиц в диапазоне от 1 до 2 (см. раздел 9). Индексы словарных таблиц алфавитов с ограниченной областью распространения в компоненте **restricted-alphabets** в **initial-vocabulary** (при его наличии) должны присваиваться следующим образом:

а) если внешний словарь отсутствует или внешний словарь содержит только встроенные алфавиты с ограниченной областью распространения, то индексы присваивают, начиная с 16;

b) в противном случае индексы присваивают, начиная с единицы плюс наибольший индекс алфавита с ограниченной областью распространения во внешнем словаре.

Примечание — Это означает, что индексы словарных таблиц от 3 до 15 не используют. Эти значения зарезервированы для последующих версий настоящего стандарта.

7.2.20 Встроенные алгоритмы кодирования имеют индексы словарных таблиц в диапазоне от 1 до 10 (см. раздел 10). Индексы словарных таблиц алгоритмов кодирования в компоненте **encoding-algorithms** в **initial-vocabulary** (при его наличии) должны присваиваться следующим образом:

a) если внешний словарь отсутствует или внешний словарь содержит только встроенные алгоритмы кодирования, то индексы присваивают, начиная с 32;

b) в противном случае индексы присваивают, начиная с единицы плюс наибольший индекс алгоритма кодирования во внешнем словаре.

Примечание — Это означает, что индексы словарных таблиц от 11 до 31 не используют. Эти значения зарезервированы для последующих версий настоящего стандарта.

7.2.21 Таблица PREFIX должна содержать встроенную запись префикса «xml», которой присвоен индекс 1. Индексы словарных таблиц префиксов в компоненте **prefixes** в **initial-vocabulary** (при его наличии) должны присваиваться следующим образом:

a) если внешний словарь отсутствует или внешний словарь содержит только встроенную запись префикса, то индексы присваивают, начиная с 2;

b) в противном случае индексы присваивают, начиная с единицы плюс наибольший индекс префикса во внешнем словаре.

7.2.22 Таблица NAMESPACE NAME должна содержать следующую встроенную запись имени пространства имен:

http://www.w3.org/XML/1998/namespace

Данной записи должен быть присвоен индекс 1.

7.2.23 Индексы словарных таблиц имен пространств имен в компоненте **namespace-names** в **initial-vocabulary** (при его наличии) должны присваиваться следующим образом:

a) если внешний словарь отсутствует или внешний словарь содержит только встроенную запись имени пространства имен, то индексы присваивают, начиная с 2;

b) в противном случае индексы присваивают, начиная с единицы плюс наибольший индекс имени пространства имен во внешнем словаре.

7.2.24 Компонент **notations** представляет свойство **[notations]** информационного элемента **document**. Типом данного компонента является sequence-of (последовательность-из), хотя свойство **[notations]** определено в инфо-наборе W3C XML как неупорядоченный набор (информационных элементов **notation**).

Примечание — Здесь и далее используют тип sequence-of (последовательность-из), а не set-of (набор-из), так как последний не удовлетворяет потребности в строгом упорядочении всех компонентов документа быстрого инфо-набора (см. 8.1).

7.2.25 Компонент **unparsed-entities** представляет свойство **[unparsed entities]** информационного элемента **document**. Типом данного компонента является sequence-of (последовательность-из), хотя свойство **[unparsed entities]** и определено в инфо-наборе W3C XML как неупорядоченный набор (информационных элементов **unparsed entity**).

7.2.26 Компонент **character-encoding-scheme** представляет свойство **[character encoding scheme]** информационного элемента **document**. Типом данного компонента является **NonEmptyOctetString**, и значение должно содержать кодирование UTF-8 (см. ISO/IEC 10646, Приложение D) свойства **[character encoding scheme]**. Отсутствие этого компонента в абстрактном значении типа **Document** указывает на то, что свойство **[character encoding scheme]** имеет значение «UTF-8».

Примечание — Поддержка свойства **[character encoding scheme]** позволяет преобразовывать документы XML в документы быстрого инфо-набора и обратно без изменения схемы кодирования символов. Создатель документа быстрого инфо-набора из документа XML может закодировать свойство **[character encoding scheme]**, полученное из декларации кодирования документа XML (см. W3C XML 1.0, п. 4.3.1 и W3C XML 1.1, п. 4.3.1). Обработчик документа быстрого инфо-набора может использовать компонент **character-encoding-scheme** (при его наличии), если хочет воспроизвести оригинальное кодирование.

7.2.27 Компонент **standalone** представляет свойство **[standalone]** информационного элемента **document**. Абстрактное значение **TRUE** представляет значение данного свойства **yes**, а абстрактное

значение **FALSE** — значение **no**. Отсутствие данного компонента в абстрактном значении типа **Document** указывает на то, что у свойства **[standalone]** нет значения.

7.2.28 Компонент **version** представляет свойство **[version]** информационного элемента **document**. Типом данного компонента является **NonIdentifyingStringOrIndex** (см. 7.14), представляющий в данном случае строку символов категории OTHER STRING. Отсутствие данного компонента в абстрактном значении типа **Document** указывает на то, что у свойства **[version]** нет значения.

7.2.29 Компонент **children** представляет свойство **[children]** информационного элемента **document**. Ровно один элемент последовательности-из (sequence-of) (в любой позиции) должен использовать альтернативу **element** выборочного типа, и не более чем один элемент (в любой позиции) должен использовать альтернативу **document-type-declaration**. Все другие элементы (при их наличии) должны использовать альтернативу **processing-instruction** или **comment**.

7.2.30 Свойство **[document element]** информационного элемента **document** не включено в тип **Document**. Значение данного свойства всегда равно только одному информационному элементу **element**, который является членом свойства **[children]** информационного элемента **document**.

7.2.31 Свойство **[base URI]** информационного элемента **document** не включено в тип **Document** и не поддерживается в настоящем стандарте.

7.2.32 Свойство **[all declarations processed]** информационного элемента **document** не включено в тип **Document** и принимается имеющим значение **true** (см. 11.3).

7.3 Тип Element

7.3.1 Тип **Element** определен следующим образом:

```
Element ::= SEQUENCE {
  namespace-attributes    SEQUENCE (SIZE(1..MAX)) OF
    NamespaceAttribute OPTIONAL,
  qualified-name          QualifiedNameOrIndex
                            -- Категория ELEMENT NAME --,
  attributes              SEQUENCE (SIZE(1..MAX)) OF
    Attribute OPTIONAL,
  children                SEQUENCE (SIZE(0..MAX)) OF
    CHOICE {
      element                          Element,
      processing-instruction           ProcessingInstruction,
      unexpanded-entity-reference      UnexpandedEntityReference,
      character-chunk                  CharacterChunk,
      comment                          Comment }}
```

7.3.2 Типы **NameSpaceAttribute**, **QualifiedNameOrIndex**, **Attribute**, **ProcessingInstruction**, **UnexpandedEntityReference**, **CharacterChunk** и **Comment** определены в 7.12, 7.16, 7.4, 7.5, 7.6, 7.7 и 7.8 соответственно.

7.3.3 Тип **Element** представляет информационный элемент **element** инфо-набора XML.

7.3.4 Компонент **namespace-attributes** представляет свойство **[namespace attributes]** информационного элемента **element**. Типом данного компонента является sequence-of (последовательность-из), хотя свойство **[namespace attributes]** определено в инфо-наборе W3C XML как неупорядоченный набор (информационных элементов **attribute**).

Примечание — Типом данного компонента является sequence-of (последовательность-из) **NamespaceAttribute** (а не **Attribute**), хотя свойство **[namespace attributes]** информационного элемента **element** определено в инфо-наборе W3C XML как набор информационных элементов **attribute**. В ограниченном инфо-наборе XML (см. 11.3) свойства информационного элемента **namespace** могут быть определены из свойств информационного элемента **attribute**, представляющего атрибут пространства имен. Обратное верно только частично, но это ограничение считается приемлемым для предполагаемого использования настоящего стандарта (см. также примечание в 7.2.24).

7.3.5 Компонент **qualified-name** представляет квалифицированное имя (см. 3.4.11) информационного элемента **element** (т. е. набор, состоящий из свойств **[prefix]**, **[namespace name]** и **[local name]** этого информационного элемента). Типом данного компонента является **QualifiedNameOrIndex** (см. 7.16), представляющий в данном случае квалифицированное имя категории ELEMENT NAME.

7.3.6 Компонент **attributes** представляет свойство **[attributes]** информационного элемента **element**. Типом данного компонента является sequence-of (последовательность-из), хотя свойство

[attributes] определено в инфо-наборе W3C XML как неупорядоченный набор (информационных элементов **attribute**).

7.3.7 Компонент **children** представляет свойство **[children]** информационного элемента **element**. Когда два или более соседних дочерних элемента являются информационными элементами **character**, для представления этих соседних информационных элементов **character** может быть использован один элемент **CharacterChunk**.

Примечание — Если среди дочерних элементов информационного элемента **element** имеется последовательность из N соседних символов, то допустима любая группировка этих N-символов в серии последовательных блоков символов. Однако предполагается, что для создания эффективных кодировок создатель документа быстрого инфо-набора будет делать каждый из блоков символов настолько большим, насколько это будет возможно.

7.3.8 Свойство **[in-scope namespaces]** информационного элемента **element** не включено в тип **Element**.

Примечание — В ограниченном инфо-наборе XML (см. 11.3) свойство **[in-scope namespaces]** информационного элемента **element** может быть определено из свойства **[namespace attributes]** информационного элемента **element** и свойства **[namespace attributes]** всех информационных элементов **element** (при их наличии), содержащих (прямо или косвенно) данный информационный элемент **element**.

7.3.9 Свойство **[base URI]** информационного элемента **element** не включено в тип **Element** и не поддерживается в настоящем стандарте.

7.3.10 Свойство **[parent]** информационного элемента **element** не включено в тип **Element**. Значением данного свойства для любого заданного информационного элемента **element** является информационный элемент **document** или **element**, содержащий заданный информационный элемент в качестве члена своего свойства **[children]**.

7.4 Тип Attribute

7.4.1 Тип **Attribute** определен следующим образом:

```
Attribute ::= SEQUENCE {
    qualified-name   QualifiedNameOrIndex
                          -- Категория ATTRIBUTE NAME --,
    normalized-value     NonIdentifyingStringOrIndex
                          -- Категория ATTRIBUTE VALUE -- }
```

7.4.2 Типы **QualifiedNameOrIndex** и **NonIdentifyingStringOrIndex** определены в 7.16 и 7.14 соответственно.

7.4.3 Тип **Attribute** представляет информационный элемент **attribute** инфо-набора XML.

7.4.4 Компонент **qualified-name** представляет квалифицированное имя (см. 3.4.11) информационного элемента **attribute** (т. е. набор, состоящий из свойств **[prefix]**, **[namespace name]** и **[local name]** этого информационного элемента). Типом данного компонента является **QualifiedNameOrIndex** (см. 7.16), представляющий в данном случае квалифицированное имя категории ATTRIBUTE NAME.

7.4.5 Компонент **normalized-value** представляет свойство **[normalized value]** информационного элемента **attribute**. Типом данного компонента является **NonIdentifyingStringOrIndex** (см. 7.14), представляющий в данном случае квалифицированное имя категории ATTRIBUTE VALUE.

7.4.6 Длина строки символов, присвоенной **normalized-value**, не может превышать 2^{32}.

Примечание — Данное ограничение обусловлено определением ASN.1, которое разработано для оптимизации кодирований и простоты реализации (см. также 11.3, перечисление j).

7.4.7 Свойство **[specified]** информационного элемента **attribute** не включено в тип **Attribute**.

7.4.8 Свойство **[attribute type]** информационного элемента **attribute** не включено в тип **Attribute**.

7.4.9 Свойство **[references]** информационного элемента **attribute** не включено в тип **Attribute**.

Примечание — В ограниченном инфо-наборе XML (см. 11.3) свойство **[references]** информационного элемента **attribute** может быть определено из свойства **[normalized value]** информационного элемента **attribute** вместе со свойствами других информационных элементов в инфо-наборе XML.

7.4.10 Свойство **[owner element]** информационного элемента **attribute** не включено в тип **Attribute**. Значение этого свойства для любого заданного информационного элемента **attribute** является информационным элементом **element**, содержащим заданный информационный элемент в качестве члена свойства **[attributes]**.

7.5 Тип ProcessingInstruction

7.5.1 Тип **ProcessingInstruction** определен следующим образом:

```
ProcessingInstruction ::= SEQUENCE {
    target   IdentifyingStringOrIndex
                                -- Категория OTHER NCNAME --,
    content  NonIdentifyingStringOrIndex
                                -- Категория OTHER STRING -- }
```

7.5.2 Типы **IdentifyingStringOrIndex** и **NonIdentifyingStringOrIndex** определены в 7.13 и 7.14 соответственно.

7.5.3 Тип **ProcessingInstruction** представляет информационный элемент **processing instruction** инфо-набора XML.

7.5.4 Компонент **target** представляет свойство **[target]** информационного элемента **processing instruction**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER NCNAME.

7.5.5 Компонент **content** представляет свойство **[content]** информационного элемента **processing instruction**. Типом данного компонента является **NonIdentifyingStringOrIndex** (см. 7.14), представляющий в данном случае строку символов категории OTHER STRING.

7.5.6 Длина строки символов, присвоенной **content**, не может превышать 2^{32}.

Примечание — Данное ограничение обусловлено определением ASN.1, которое разработано для оптимизации кодирований и простоты реализации (см. также 11.3, перечисление j).

7.5.7 Свойство **[notation]** информационного элемента **processing instruction** не включено в тип **ProcessingInstruction**.

Примечание — В ограниченном инфо-наборе XML (см. 11.3) свойство **[notation]** информационного элемента **processing instruction** может быть определено из свойства **[target]** информационного элемента **processing instruction** и свойства **[notations]** информационного элемента **document**.

7.5.8 Свойство **[parent]** информационного элемента **processing instruction** не включено в тип **ProcessingInstruction**. Значением этого свойства для любого заданного информационного элемента **processing instruction** является информационный элемент **document**, **element** или **document type definition**, содержащий заданный информационный элемент в качестве члена свойства **[children]**.

7.6 Тип UnexpandedEntityReference

7.6.1 Тип **UnexpandedEntityReference** определен следующим образом:

```
UnexpandedEntityReference ::= SEQUENCE {
    name                    IdentifyingStringOrIndex
                                -- Категория OTHER NCNAME --,
    system-identifier    IdentifyingStringOrIndex OPTIONAL
                                -- Категория OTHER URI --,
    public-identifier    IdentifyingStringOrIndex OPTIONAL
                                -- Категория OTHER URI -- }
```

7.6.2 Тип **IdentifyingStringOrIndex** определен в 7.13.

7.6.3 Тип **UnexpandedEntityReference** представляет информационный элемент **unexpanded entity reference** инфо-набора XML.

7.6.4 Компонент **name** представляет свойство **[name]** информационного элемента **unexpanded entity reference**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER NCNAME.

7.6.5 Компонент **system-identifier** представляет свойство **[system identifier]** информационного элемента **unexpanded entity reference**. Типом данного компонента является **IdentifyingStringOrIndex** (см 7.13), представляющий в данном случае строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **UnexpandedEntityReference** указывает на то, что свойство **[system identifier]** не имеет значения.

7.6.6 Компонент **public-identifier** представляет свойство **[public identifier]** информационного элемента **unexpanded entity reference**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER URI.

Отсутствие данного компонента в абстрактном значении типа **UnexpandedEntityReference** указывает на то, что свойство **[public identifier]** не имеет значения.

7.6.7 Свойство **[declaration base URI]** информационного элемента **unexpanded entity reference** не включено в тип **UnexpandedEntityReference** и не поддерживается в настоящем стандарте.

7.6.8 Свойство **[parent]** информационного элемента **unexpanded entity reference** не включено в тип **UnexpandedEntityReference**. Значением данного свойства для любого заданного информационного элемента **unexpanded entity reference** является информационный элемент **element**, содержащий заданный информационный элемент в качестве члена свойства **[children]**.

7.7 Тип CharacterChunk

7.7.1 Тип **CharacterChunk** определен следующим образом:

```
CharacterChunk ::= SEQUENCE {
  character-codes   NonIdentifyingStringOrIndex
                        -- Категория CONTENT CHARACTER CHUNK -- }
```

7.7.2 Тип **NonIdentifyingStringOrIndex** определен в 7.14.

7.7.3 Тип **CharacterChunk** соответствует информационному элементу **character**, но представляет ряд соседних информационных элементов **character** (членов свойства **[children]** родительского информационного элемента **element**), а не единственный информационный элемент **character**.

7.7.4 Число информационных элементов **character**, представленных значением типа **CharacterChunk**, не должно быть равным нулю.

7.7.5 Компонент **character-codes** представляет свойство **[character code]** информационного(ых) элемента(ов) **character** в блоке. Типом данного компонента является **NonIdentifyingStringOrIndex** (см. 7.14), представляющий в данном случае строку символов категории CONTENT CHARACTER CHUNK.

7.7.6 Длина строки символов, присвоенной **character-codes**, не может превышать 2^{32}.

Примечание — Данное ограничение обусловлено определением ASN.1, которое разработано для оптимизации кодирований и простоты реализации. Данное ограничение не исключает кодирования информационного элемента **element**, содержащего более 2^{32} информационных элементов **character**, так как можно использовать несколько блоков.

7.7.7 Свойство **[element content whitespace]** информационного(ых) элемента(ов) **character** не включено в тип **CharacterChunk**.

7.7.8 Свойство **[parent]** информационного(ых) элемента(ов) **character** не включено в тип **CharacterChunk**. Значением данного свойства для любого заданного информационного элемента **character** является информационный элемент **element**, содержащий заданный информационный элемент в качестве члена свойства **[children]**.

7.8 Тип Comment

7.8.1 Тип **Comment** определен следующим образом:

```
Comment ::= SEQUENCE {
     content          NonIdentifyingStringOrIndex
                              -- Категория OTHER STRING --}
```

7.8.2 Тип **NonIdentifyingStringOrIndex** определен в 7.14.

7.8.3 Тип **Comment** представляет информационный элемент **comment** инфо-набора XML.

7.8.4 Компонент **content** представляет свойство **[content]** информационного элемента **comment**. Типом данного компонента является **NonIdentifyingStringOrIndex** (см. 7.14), представляющий в данном случае строку символов категории OTHER STRING.

7.8.5 Длина строки символов, присвоенной **content**, не может превышать 2^{32}.

Примечание — Данное ограничение обусловлено определением ASN.1, которое разработано для оптимизации кодирований и простоты реализации (см. также 11.3, перечисление j).

7.8.6 Свойство **[parent]** информационного элемента **comment** не включено в тип **Comment**. Значением этого свойства для любого заданного информационного элемента **comment** является информационный элемент **document** или **element**, содержащий заданный информационный элемент в качестве члена свойства **[children]**.

7.9 Тип DocumentTypeDeclaration

7.9.1 Тип **DocumentTypeDeclaration** определен следующим образом:

```
DocumentTypeDeclaration ::= SEQUENCE {
     system-identifier     IdentifyingStringOrIndex OPTIONAL
                                   -- Категория OTHER URI --,
     public-identifier     IdentifyingStringOrIndex OPTIONAL
                                   -- Категория OTHER URI --,
     children                   SEQUENCE (SIZE(0..MAX)) OF
          ProcessingInstruction }
```

7.9.2 Тип **IdentifyingStringOrIndex** определен в 7.13.

7.9.3 Тип **DocumentTypeDeclaration** представляет информационный элемент **document type declaration** инфо-набора XML.

7.9.4 Компонент **system-identifier** представляет свойство **[system identifier]** информационного элемента **document type declaration**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **DocumentTypeDeclaration** указывает на то, что свойство **[system identifier]** не имеет значения.

7.9.5 Компонент **public-identifier** представляет свойство **[public identifier]** информационного элемента **document type declaration**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **DocumentTypeDeclaration** указывает на то, что свойство **[public identifier]** не имеет значения.

7.9.6 Компонент **children** представляет свойство **[children]** информационного элемента **document type declaration**.

7.9.7 Свойство **[parent]** информационного элемента **document type declaration** не включено в тип **DocumentTypeDeclaration**. Значением этого свойства для любого заданного информационного элемента **document type declaration** является информационный элемент **document**, содержащий заданный информационный элемент в качестве члена свойства **[children]**.

7.10 Тип UnparsedEntity

7.10.1 Тип **UnparsedEntity** определен следующим образом:

```
UnparsedEntity ::= SEQUENCE {
   name                    IdentifyingStringOrIndex
                        -- Категория OTHER NCNAME --,
   system-identifier    IdentifyingStringOrIndex
                        -- Категория OTHER URI --,
   public-identifier    IdentifyingStringOrIndex OPTIONAL
                        -- Категория OTHER URI --,
   notation-name           IdentifyingStringOrIndex
                        -- Категория OTHER NCNAME -- }
```

7.10.2 Тип **IdentifyingStringOrIndex** определен в 7.13.

7.10.3 Тип **UnparsedEntity** представляет информационный элемент **unparsed entity** инфо-набора XML.

7.10.4 Компонент **name** представляет свойство **[name]** информационного элемента **unparsed entity**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER NCNAME.

7.10.5 Компонент **system-identifier** представляет свойство **[system identifier]** информационного элемента **unparsed entity**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER URI.

7.10.6 Компонент **public-identifier** представляет свойство **[public identifier]** информационного элемента **unparsed entity**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **UnparsedEntity** указывает на то, что свойство **[public identifier]** не имеет значения.

7.10.7 Компонент **notation-name** представляет свойство **[notation name]** информационного элемента **unparsed entity**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER NCNAME.

7.10.8 Свойство **[declaration base URI]** информационного элемента **unparsed entity** не включено в тип **UnparsedEntity** и не поддерживается в настоящем стандарте.

7.10.9 Свойство **[notation]** информационного элемента **unparsed entity** не включено в тип **UnparsedEntity**.

Примечание — В ограниченном инфо-наборе XML (см. 11.3) свойство **[notation]** информационного элемента **unparsed entity** может быть определено из свойства **[notation name]** информационного элемента **unparsed entity** и свойства **[notations]** информационного элемента **document**.

7.11 Тип Notation

7.11.1 Тип **Notation** определен следующим образом:

```
Notation ::= SEQUENCE {
     name                        IdentifyingStringOrIndex
                                      -- Категория OTHER NCNAME --,
     system-identifier      IdentifyingStringOrIndex OPTIONAL
                                      -- Категория OTHER URI --,
     public-identifier      IdentifyingStringOrIndex OPTIONAL
                                      -- Категория OTHER URI -- }
```

7.11.2 Тип **IdentifyingStringOrIndex** определен в 7.13.

7.11.3 Тип **Notation** представляет информационный элемент **notation** инфо-набора XML.

7.11.4 Компонент **name** представляет свойство **[name]** информационного элемента **notation**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER NCNAME.

7.11.5 Компонент **system-identifier** представляет свойство **[system identifier]** информационного элемента **notation**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **Notation** указывает на то, что свойство **[system identifier]** не имеет значения.

7.11.6 Компонент **public-identifier** представляет свойство **[public identifier]** информационного элемента **notation**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **Notation** указывает на то, что свойство **[public identifier]** не имеет значения.

7.11.7 Свойство **[declaration base URI]** информационного элемента **notation** не включено в тип **Notation** и не поддерживается в настоящем стандарте.

7.12 Тип NamespaceAttribute

7.12.1 Тип **NamespaceAttribute** определен следующим образом:

```
NamespaceAttribute ::= SEQUENCE {
     prefix           IdentifyingStringOrIndex OPTIONAL
                           -- Категория PREFIX --,
     namespace-name        IdentifyingStringOrIndex OPTIONAL
                           -- Категория NAMESPACE NAME -- }
```

7.12.2 Тип **IdentifyingStringOrIndex** определен в 7.13.

7.12.3 Тип **NamespaceAttribute** представляет информационный элемент **attribute**, который является членом свойства **[namespace attributes]** информационного элемента **element** инфо-набора XML.

Примечание — В инфо-наборе XML атрибуты и пространства имен являются информационными элементами **attribute**. В настоящем стандарте использованы разные типы с целью оптимизации.

7.12.4 В инфо-наборе XML имеется два типа атрибутов пространств имен:

a) принимаемые по умолчанию декларации пространств имен: свойство **[prefix]** информационного элемента **attribute** не имеет значения, а значением свойства **[local name]** является «xmlns»;

b) принимаемые не по умолчанию декларации пространств имен: свойство **[prefix]** информационного элемента **attribute** имеет значение «xmlns», а свойство **[local name]** предоставляет префикс декларации пространства имен.

В обоих случаях свойство **[normalized value]** информационного элемента **attribute** предоставляет имя пространства имен декларации пространства имен.

7.12.5 Если атрибут пространства имен является принимаемой по умолчанию декларацией пространства имен (7.12.4, перечисление a), то компонент **prefix** должен отсутствовать, в противном случае (7.12.4, перечисление b) он должен присутствовать, представляя свойство **[local name]** информационного элемента **attribute**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории PREFIX.

7.12.6 Если свойство **[normalized value]** информационного элемента **attribute** является пустой строкой, то компонент **namespace-name** должен отсутствовать, в противном случае он должен присутствовать, представляя свойство **[normalized value]** информационного элемента **attribute**. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории NAMESPACE NAME.

7.12.7 Свойство **[namespace name]** информационного элемента **attribute** всегда имеет значение «» (см. инфо-набор W3C XML) и не включено в тип **NamespaceAttribute**.

gost.e-ecolog.ru

7.13 Тип IdentifyingStringOrIndex

7.13.1 Тип **IdentifyingStringOrIndex** определен следующим образом:

```
IdentifyingStringOrIndex ::= CHOICE {
     literal-character-string          NonEmptyOctetString,
     string-index                      INTEGER (1..one-meg) }
```

7.13.2 Тип **NonEmptyOctetString** и значение **one-meg** определены в 7.2.1.

7.13.3 Тип **IdentifyingStringOrIndex** представляет строку символов, несущую идентификационную информацию.

Примечание — Примерами таких строк символов являются префиксы, имена пространств имен и локальные имена элементов и атрибутов.

7.13.4 Абстрактное значение этого типа ASN.1 содержит либо строку символов (заданной категории) в качестве значения типа **NonEmptyOctetString**, либо индекс словарной таблицы строки символов (заданной категории) в словарной таблице для данной категории строк (см. 8.4.2), которую называют применяемой таблицей строк.

Примечания

1 При использовании данного типа ASN.1 категория строки всегда специфицирована в соответствующем тексте в предыдущих пунктах (см. 7.5—7.12).

2 Идентифицирующие строки символов трактуются не так, как неидентифицирующие. Неидентифицирующие строки символов могут быть закодированы в одном из многих форматов кодирования, а все идентифицирующие строки символов должны быть закодированы в UTF-8. Кроме этого, неидентифицирующие строки символов могут добавляться или не добавляться (по выбору создателя) в динамические таблицы строк (см. 7.14.6), а идентифицирующие строки символов всегда добавляются в динамические таблицы строк (см. 7.13.7).

7.13.5 Компонент **literal-character-string** (при его наличии) должен содержать кодирование UTF-8 (см. ISO/IEC 10646, Приложение D) строки символов (см. 7.13.4).

7.13.6 Компонент **string-index** (при его наличии) должен содержать индекс словарной таблицы какой-либо из идентичных данной строке символов записей применяемой таблицы строк.

7.13.7 Когда создают абстрактное значение этого типа ASN.1 (представляющее заданную строку символов заданной категории), то в случае, если идентичная строка символов существует в текущем содержимом применяемой таблицы строк, создатель документа быстрого инфо-набора должен осуществить одно из перечисленных ниже действий a) или b) в зависимости от реализации (но по мере возможности следует выбирать первый вариант, так как при этом создается меньше индексов, указывающих на одну и ту же строку символов), в противном случае (не существует идентичной строки символов) создатель документа быстрого инфо-набора должен действовать согласно перечислению b). Действия a) и b) приведены ниже:

a) выбрать альтернативу **string-index** и присвоить **string-index** словарной таблицы любой из существующих записей, идентичных данной строке символов;

b) выбрать альтернативу **literal-character-string**, присвоить заданную строку символов **literal-character-string** и добавить в применяемую таблицу строк идентичную строку символов, если данная таблица не содержит 2^{20} записей.

Примечание — Выбор действия b) приведет к появлению нескольких идентичных строк символов в таблице строк (если таблица еще не содержит 2^{20} записей). Это не влияет на последующую обработку строк символов (см. 7.13.8).

7.13.8 При обработке абстрактного значения данного типа ASN.1, представляющего строку символов (заданной категории), обработчик документа быстрого инфо-набора должен определить строку символов, представленную абстрактным значением, следующим образом:

a) если имеется альтернатива **string-index**, то строка символов, представленная абстрактным значением, должна быть строкой символов в текущем содержимом применяемой таблицы строк, индекс словарной таблицы которой является значением **string-index**;

b) если имеется альтернатива **literal-character-string**, то строка символов, представленная абстрактным значением, должна быть значением **literal-character-string**, а идентичная строка символов должна быть добавлена к применяемой таблице строк (см. 7.13.9), когда данная таблица еще не содержит 2^{20} записей.

Примечание — Выбор действия b) приведет к появлению нескольких идентичных строк символов в таблице строк (если таблица еще не содержит 2^{20} записей). Это не влияет на последующую обработку строк символов.

7.13.9 Если обработчик документа быстрого инфо-набора не может (по какой-либо причине, включая зависящие от реализации ограничения) добавить строку в словарную таблицу, содержащую менее 2^{20} записей, когда такое добавление требуется согласно 7.13.8, перечисление b), то он должен прекратить обработку документа быстрого инфо-набора и сообщить об ошибке.

7.14 Тип NonIdentifyingStringOrIndex

7.14.1 Тип **NonIdentifyingStringOrIndex** определен следующим образом:

```
NonIdentifyingStringOrIndex ::= CHOICE {
     literal-character-string SEQUENCE {
          add-to-table                BOOLEAN,
          character-string            EncodedCharacterString },
     string-index               INTEGER (0..one-meg) }
```

7.14.2 Тип **EncodedCharacterString** и значение **one-meg** определены в 7.17 и 7.2.1 соответственно.

7.14.3 Тип **NonIdentifyingStringOrIndex** представляет строку символов, не несущую идентификационную информацию.

Примечание — Примером такой строки символов служит значение атрибута.

7.14.4 Абстрактное значение типа **NonIdentifyingStringOrIndex** содержит либо строку символов (заданной категории) в качестве значения типа **EncodedCharacterString** (см. 7.17), либо индекс словарной таблицы строки символов заданной категории в словарной таблице для этой категории строк (см. 8.4.2), которую называют применяемой таблицей строк.

Примечание — При использовании данного типа категория строки всегда специфицирована в соответствующем тексте в предыдущих пунктах.

7.14.5 Компонент **string-index** (при его наличии) должен либо иметь значение, равное нулю (обозначая строку символов нулевой длины — см. 7.14.6), либо содержать индекс словарной таблицы какой-либо из записей применяемой таблицы строк, которая идентична данной строке символов.

7.14.6 Для строки символов нулевой длины создатель документа быстрого инфо-набора всегда должен использовать альтернативу **string-index** с целым значением, равным нулю. Обработчик документа быстрого инфо-набора должен трактовать такое значение как представление строки символов нулевой длины.

7.14.7 Когда создают абстрактное значение типа **NonIdentifyingStringOrIndex** (представляющее заданную строку символов заданной категории) ненулевой длины, то в случае, если идентичная строка символов существует в текущем содержимом применяемой таблицы строк, создатель документа быстрого инфо-набора должен осуществить одно из перечисленных ниже действий a) или b) в зависимости от реализации (но по мере возможности следует выбирать первый вариант, так как при этом создается меньше индексов, указывающих на одну и ту же строку символов), в противном случае (не существует идентичной строки символов) создатель документа быстрого инфо-набора должен действовать согласно перечислению b). Действия a) и b) приведены ниже:

a) выбрать альтернативу **string-index** и присвоить **string-index** индекс любой из существующих записей словарной таблицы, идентичных данной строке символов;

b) выбрать альтернативу **literal-character-string**, присвоить заданную строку символов **literal-character-string** и:

1) добавить в применяемую таблицу строк идентичную строку символов и установить компонент **add-to-table** равным **TRUE** [данное действие b1) не должно использоваться в случае, если применяемая таблица строк уже содержит 2^{20} записей] или

Примечание — Если применяемая таблица строк уже содержит 2^{20} записей, то возможны только действия a) или b2).

2) установить компонент **add-to-table** равным **FALSE**.

Примечание — Выбор действия b1) приведет к появлению нескольких идентичных строк символов в текущем содержимом. Это не влияет на последующую обработку строк символов (см. 7.14.8).

7.14.8 При обработке абстрактного значения этого типа ASN.1, представляющего строку символов заданной категории, обработчик документа быстрого инфо-набора должен определить строку символов, представленную абстрактным значением, следующим образом:

a) если имеется альтернатива **string-index**, то строка символов, представленная абстрактным значением, должна быть строкой символов в применяемой таблице строк, индекс словарной таблицы которой является значением **string-index**.

Примечание 1 — Если значение **string-index** превышает текущий размер этой словарной таблицы, то документ быстрого инфо-набора ошибочен;

b) если имеется альтернатива **literal-character-string** и компонент **add-to-table** имеет значение **TRUE**, то строка символов, представленная абстрактным значением, должна быть значением **literal-character-string**. Обработчик документа быстрого инфо-набора должен добавить идентичную строку символов к применяемой таблице строк (см. 7.14.9).

Примечание 2 — Если применяемая таблица строк уже содержит 2^{20} строк, то документ быстрого инфо-набора ошибочен;

c) если имеется альтернатива **literal-character-string** и компонент **add-to-table** имеет значение **FALSE**, то строка символов, представленная данным абстрактным значением, должна быть значением компонента **character-string**.

7.14.9 Если обработчик документа быстрого инфо-набора не может (по какой-либо причине, включая зависящие от реализации ограничения) добавить строку в словарную таблицу, когда такое добавление требуется согласно 7.14.8, перечисление b), то он должен прекратить обработку документа быстрого инфо-набора и сообщить об ошибке.

7.15 Тип NameSurrogate

7.15.1 Тип **NameSurrogate** определен следующим образом:

```
NameSurrogate ::= SEQUENCE {
   prefix-string-index                INTEGER(1..one-meg) OPTIONAL,
   namespace-name-string-index  INTEGER(1..one-meg) OPTIONAL,
   local-name-string-index            INTEGER(1..one-meg) }
   (CONSTRAINED BY {-- prefix-string-index должен присутствовать,
-- только если присутствует
-- namespace-name-string-index --})
```

7.15.2 Значение **one-meg** определено в 7.2.1.

7.15.3 Тип **NameSurrogate** содержит три положительных целых значения (первые два являются необязательными), образующих идентификатор имени (см. 8.5.2).

Примечание — Данный тип встречается только в компоненте **initial-vocabulary** типа **Document**.

7.15.4 Компоненты **prefix-string-index** (при его наличии), **namespace-name-string-index** (при его наличии) и **local-name-string-index** должны быть больше нуля и не больше числа записей в таблицах PREFIX, NAMESPACE NAME и LOCAL-NAME исходного словаря соответственно.

Примечание — Если обработчик документа быстрого инфо-набора решит не проверять выполнение данного ограничения, то в результате дальнейшей обработки могут возникнуть уязвимости, связанные с безопасностью.

7.15.5 Компонент **prefix-string-index** должен отсутствовать, если отсутствует компонент **namespace-name-string-index**.

7.16 Тип QualifiedNameOrIndex

7.16.1 Тип **QualifiedNameOrIndex** определен следующим образом:

```
QualifiedNameOrIndex ::= CHOICE {
    literal-qualified-name      SEQUENCE {
        prefix                   IdentifyingStringOrIndex OPTIONAL
                                   -- Категория PREFIX --,
        namespace-name    IdentifyingStringOrIndex OPTIONAL
                                   -- Категория NAMESPACE NAME --,
        local-name               IdentifyingStringOrIndex
                                   -- Категория LOCAL NAME -- },
    name-surrogate-index INTEGER (1..one-meg) }
```

7.16.2 Тип **IdentifyingStringOrIndex** и значение **one-meg** определены в 7.13 и 7.2.1 соответственно.

7.16.3 Тип **QualifiedNameOrIndex** представляет квалифицированное имя (см. 3.4.11).

7.16.4 Абстрактное значение данного типа ASN.1 содержит либо три компонента, соответствующие префиксу, имени пространства имен и локальному имени квалифицированного имени (заданной категории), либо индекс словарной таблицы идентификатора имени заданной категории в словарной таблице для этой категории квалифицированных имен (см. 8.5.2), которую называют применяемой таблицей имен.

Примечание — При использовании данного типа категория всегда специфицирована в соответствующем тексте в предыдущих пунктах.

7.16.5 Компонент **name-surrogate-index** (при его наличии) должен содержать индекс словарной таблицы идентификатора имени в применяемой таблице имен.

7.16.6 Если компонент **namespace-name** отсутствует, то компонент **prefix** также должен отсутствовать.

7.16.7 При создании абстрактного значения этого типа ASN.1, представляющего заданное квалифицированное имя (заданной категории), создатель документа быстрого инфо-набора должен выполнять действия, определенные в последующих подпунктах.

7.16.7.1 Должна быть выполнена проверка следующих условий в указанном порядке:

a) квалифицированное имя не имеет префикса или префикс квалифицированного имени существует в текущем содержимом таблицы PREFIX;

b) квалифицированное имя не имеет имени пространства имен или имя пространства имен квалифицированного имени существует в текущем содержимом таблицы NAMESPACE NAME;

c) локальное имя квалифицированного имени существует в текущем содержимом таблицы LOCAL NAME;

d) первые три условия выполнены, и идентификатор имени (см. 8.5), состоящий из индекса(ов) словарной таблицы префиксов (при его наличии), имени пространства имен (при его наличии) и локального имени, существует в текущем содержимом применяемой таблицы имен.

7.16.7.2 Если все перечисленные выше условия выполнены, то должна быть выбрана альтернатива **name-surrogate-index**, и она должна быть установлена равной индексу словарной таблицы идентификатора имени, определенному в 7.16.7.1, перечисление d), в применяемой таблице имен, завершая процедуры 7.16.7.

7.16.7.3 В противном случае должна быть выбрана альтернатива **literal-qualified-name**, и ее компонентам должны быть присвоены следующие значения:

a) если квалифицированное имя не имеет префикса, то компонент **prefix** должен отсутствовать, в противном случае компоненту **prefix** должен быть присвоен префикс по 7.13.7 с ограничением, что действие 7.13.7, перечисление b) не должно выполняться, если идентичная строка символов существует в текущем содержимом применяемой таблицы строк. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории PREFIX;

b) если квалифицированное имя не имеет имени пространства имен, то компонент **namespace-name** должен отсутствовать, в противном случае компоненту **namespace-name** должно быть присвоено имя пространства имен по 7.13.7 с ограничением, что действие 7.13.7, перечисление b) не должно выполняться, если идентичная строка символов существует в текущем содержимом применяемой таблицы строк. Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории NAMESPACE NAME (см. 8.4.2);

c) компоненту **local-name** должно быть присвоено локальное имя квалифицированного имени (по 7.13.7). Типом данного компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории LOCAL NAME (см. 8.4.2).

Примечание — Выполнение действий согласно 7.13.7 в настоящем подпункте может вызывать добавление локального имени в таблицу LOCAL NAME.

7.16.7.4 Если выполнение действий согласно 7.13.7 в подпункте 7.16.7.3 к одной или нескольким из трех указанных выше строк символов не привело к добавлению строки в словарную таблицу в связи с тем, что в таблица уже содержит 2^{20} записей [см. 7.13.7, перечисление b)], то идентификатор имени для данного квалифицированного имени не может быть создан.

7.16.7.5 В противном случае должен быть создан идентификатор имени (см. 8.5), состоящий из индекса(ов) словарных таблиц префикса (при его наличии), имени пространства имен (при его наличии) и локального имени. Если этот идентификатор имени не существует в текущем содержимом применяемой таблицы имен, то он должен быть добавлен в эту таблицу, если только она уже не содержит 2^{20} записей.

7.16.8 При обработке абстрактного значения типа **QualifiedNameOrIndex**, представляющего квалифицированное имя заданной категории, обработчик документа быстрого инфо-набора должен определить квалифицированное имя, представленное абстрактным значением, в соответствии с последующими подпунктами.

7.16.8.1 Если имеется альтернатива **name-surrogate-index**, то квалифицированное имя, представленное абстрактным значением, должно быть именем, представленным идентификатором имени заданной категории (см. 8.5.2) в применяемой таблице имен, индекс словарной таблицы которого равен значению **name-surrogate-index**.

7.16.8.2 Если имеется альтернатива **literal-qualified-name**, то:

a) квалифицированное имя, представленное абстрактным значением, должно быть определено следующим образом:

1) префикс квалифицированного имени должен быть определен (по 7.13.8) из компонента **prefix** (при его наличии); типом этого компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории PREFIX;

2) имя пространства имен квалифицированного имени должно быть определено (по 7.13.8) из компонента **namespace-name** (при его наличии); типом этого компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории NAMESPACE NAME (см. 8.4.2);

3) локальное имя квалифицированного имени должно быть определено (по 7.13.8) из компонента **local-name**; типом этого компонента является **IdentifyingStringOrIndex** (см. 7.13), представляющий в данном случае строку символов категории LOCAL NAME (см. 8.4.2).

Примечание — Указанные действия могут потребовать добавления в соответствующие словарные таблицы согласно 7.13.8, перечисление b);

b) если после возможных добавлений в результате обработки компонентов **literal-qualified-name** индексы доступны для всех присутствующих компонентов, то идентификатор имени, состоящий из этих индексов словарных таблиц, должен быть добавлен в применяемую таблицу имен, если только она уже не содержит 2^{20} идентификаторов имен (см. 7.16.9).

7.16.9 Если обработчик документа быстрого инфо-набора не может (по какой-либо причине, включая связанные с реализацией ограничения) добавить идентификатор имени в словарную таблицу, содержащую менее 2^{20} записей, когда такое добавление требуется по 7.16.8.2, перечисление b), то он должен прекратить обработку документа быстрого инфо-набора и сообщить об ошибке.

7.17 Тип EncodedCharacterString

7.17.1 Тип **EncodedCharacterString** определен следующим образом:

```
EncodedCharacterString ::= SEQUENCE {
    encoding-format      CHOICE {
        utf-8                  NULL,
        utf-16                 NULL,
        restricted-alphabet INTEGER(1..256),
        encoding-algorithm     INTEGER(1..256) },
    octets               NonEmptyOctetString }
```

7.17.2 Тип **EncodedCharacterString** содержит кодирование строки символов. Он определяет строку октетов, которая является обратимым отображением строки символов в строку октетов. Создатель документа быстрого инфо-набора определяет в **encoding-format** использованное кодирование, а обработчик документа быстрого инфо-набора использует эту информацию для декодирования октетов в компоненте **octets** в строку символов.

7.17.3 Компонент **octets** передает значение строки октетов, которая является кодированием строки символов, заданным в компоненте **encoding-format**.

Примечание — В общем случае существует несколько кодирований, которые могут быть использованы для данной строки символов. Создатель документа быстрого инфо-набора может выбрать кодирование на основании определенных критериев (например, он может постараться оптимизировать размер или скорость обработки), но может предпочесть удобство и универсальную применимость UTF-8 или UTF-16BE. Возможны случаи, когда некоторые кодирования будут применимы только для строк символов, которые содержат только подмножество символов ISO/IEC 10646.

7.17.4 Формат кодирования **utf-8** может быть использован для любой строки символов. Данный формат кодирования следует применять, создавая кодирование UTF-8 (см. ISO/IEC 10646) строки символов и присваивая его компоненту **octets**.

Примечание — Этот формат кодирования наиболее подходит для строк символов, в которых наиболее часто встречаются символы из начальной части основной многоязычной плоскости ISO/IEC 10646 и когда нет других применимых или более полезных форматов кодирования.

7.17.5 Формат кодирования **utf-16** может быть использован для любой строки символов. Данный формат кодирования следует применять, создавая кодирование UTF-16BE (см. Unicode, п. 2.6) строки символов и присваивая его компоненту **octets**.

Примечания

1 Этот формат кодирования наиболее подходит для строк символов, в которых присутствует широкий диапазон символов ISO/IEC 10646 и когда нет других применимых или более полезных форматов кодирования.

2 Порядок байтов в кодировании UTF-16BE определен в Unicode, п. 2.6: самый старший байт первый (наиболее ранний байт в строке октетов).

7.17.6 Формат кодирования **restricted-alphabet** основан на использовании алфавита с ограниченной областью распространения, выбираемого из имеющихся в таблице алфавитов с ограниченной областью распространения. Компонент **restricted-alphabet** должен содержать индекс словарной таблицы алфавитов с ограниченной областью распространения. Это кодирование может быть использовано только для строки символов, которая полностью состоит из символов алфавита с ограниченной областью распространения в записи таблицы алфавитов с ограниченной областью распространения, проиндексированной компонентом **restricted-alphabet**. Формат кодирования **restricted-alphabet** следует применять, как определено в 7.17.6.1—7.17.6.6.

7.17.6.1 Каждому символу алфавита с ограниченной областью распространения должно быть присвоено целое значение (начиная с нуля) по порядку.

7.17.6.2 Каждый символ в строке символов должен быть преобразован в целое значение, которое присвоено данному символу в алфавите с ограниченной областью распространения.

7.17.6.3 Каждое целое значение должно быть представлено как целое двоичное значение без знака в битовом поле. Размер битового поля должен определяться целым значением, присвоенным последнему символу в алфавите с ограниченной областью распространения. Это целое значение должно быть увеличено на 1, чтобы получить другое целое значение (например, N). Размер битового поля должен быть равен минимальному количеству битов, необходимых для кодирования N как целого значения без знака.

Примечания

1 Увеличение на 1 необходимо потому, что значение из одних единиц в битовом поле интерпретируется как конец строки символов и не может быть использовано для представления символа. Это означает, что если алфавит с ограниченной областью распространения содержит некоторое число символов, точно равное степени двух, то битовое поле должно содержать на один бит больше, чем можно ожидать в другом случае.

2 Например, если в алфавите с ограниченной областью распространения содержится 24 символа, то каждый из символов будет кодироваться в 5 битов, но если в алфавите 32 символа, то каждый из них будет кодироваться в 6 битов.

7.17.6.4 Все полученные битовые поля должны быть сцеплены (по порядку) в битовую строку.

7.17.6.5 Если длина полученной битовой строки не кратна 8, то в конец строки должны быть добавлены биты 1 до длины, кратной 8.

7.17.6.6 Полученная битовая строка (кратная 8 битам), интерпретируемая как строка октетов, должна быть присвоена компоненту **octets**.

7.17.7 Формат кодирования **encoding-algorithm** определяется алгоритмом кодирования (см. 8.3), который содержится в записи таблицы алгоритмов кодирования, индекс словарной таблицы для которой равен значению компонента **encoding-algorithm**. Индекс словарной таблицы алгоритмов кодирования должен быть присвоен компоненту **encoding-algorithm**, полученная в результате кодирования строка октетов должна быть присвоена компоненту **octets**.

8 Построение и обработка документа быстрого инфо-набора

В документе быстрого инфо-набора используют индексы различных словарных таблиц, создаваемых на разных этапах построения и обработки этого документа. В 8.1 определен концептуальный порядок компонентов абстрактного значения типа **Document**, способствующий тому, что создатель и обработчики документа быстрого инфо-набора создают идентичные словарные таблицы. В последующих подразделах определены словарные таблицы, которые создают и используют при создании и обработке документа быстрого инфо-набора. Представление данных таблиц в компьютерной системе зависит от реализации и не стандартизировано. Словарная таблица обеспечивает отображение из индекса словарной таблицы в информацию инфо-набора XML (возможно, косвенно).

Примечание — Словарные таблицы для документа быстрого инфо-набора создают при построении документа быстрого инфо-набора. Их создают динамически из содержимого документа быстрого инфо-набора при обработке данного документа. Обмен таблицами в какой-либо форме не проводят.

8.1 Концептуальный порядок компонентов абстрактного значения типа Document

8.1.1 Концептуальный порядок компонентов абстрактного значения типа **Document** определен для того, чтобы разные реализации присваивали индексы словарных таблиц одним и тем же способом при создании и обработке документа быстрого инфо-набора. При построении и обработке таких абстрактных значений следует использовать данный концептуальный порядок, например, при добавлении строк (см. 7.13.7 и 7.14.6) и идентификаторов имен (см. 7.16.7) в словарные таблицы.

Примечание — Данный порядок совпадает с порядком кодирований компонентов в документе быстрого инфо-набора. Это необязательно подразумевает, что семантика, передаваемая документом, обрабатывается в том же порядке. Указанный порядок определен исключительно в целях обеспечения того, что один и тот же индекс словарной таблицы присвоен данной записи словарной таблицы как создателем, так и обработчиком документа быстрого инфо-набора.

8.1.2 Концептуальный порядок для построения и обработки документа быстрого инфо-набора определен следующим образом: компоненты абстрактного значения типа **Document** должны быть рассмотрены в соответствии с алгоритмом, установленным в 8.1.2.1—8.1.2.5. Порядок, в котором рассматривают компоненты, определяет концептуальный порядок.

8.1.2.1 Компонент верхнего уровня абстрактного значения (соответствующий типу **Document**) должен быть рассмотрен первым.

8.1.2.2 Если рассматриваемый компонент относится к типу sequence (последовательность), то компоненты данного типа, присутствующие в абстрактном значении, следует рассмотреть в порядке их определения по тексту — от первого присутствующего компонента до последнего, выполняя для каждого рассматриваемого компонента действия 8.1.2.1—8.1.2.5 рекурсивно.

8.1.2.3 Если рассматриваемый компонент относится к типу sequence-of (последовательность-из), то все появления компонента sequence-of следует рассмотреть в порядке sequence-of — от первого появления компонента до последнего, выполняя для каждого рассматриваемого компонента действия 8.1.2.1—8.1.2.5 рекурсивно.

8.1.2.4 Если рассматриваемый компонент относится к типу choice (выбор), то следует рассмотреть альтернативу, присутствующую в абстрактном значении, выполняя для данной альтернативы действия 8.1.2.1—8.1.2.5 рекурсивно.

8.1.2.5 Если рассматриваемый компонент относится к какому-либо иному типу ASN.1, то никаких дальнейших действий для данного компонента не требуется.

8.2 Таблица алфавитов с ограниченной областью распространения

8.2.1 Каждый документ быстрого инфо-набора имеет связанную с ним таблицу алфавитов с ограниченной областью распространения. Таблица алфавитов с ограниченной областью распространения содержит алфавиты с ограниченной областью распространения, на которые можно ссылаться через индекс словарной таблицы.

8.2.2 Каждая запись в таблице алфавитов с ограниченной областью распространения должна быть упорядоченным набором разных символов по ISO/IEC 10646 размером от 2 до 2^{20} символов.

Примечание — Алфавит с ограниченной областью распространения допускает компактное кодирование любой строки символов, полностью состоящей из символов этого набора, путем присвоения по нарастанию целых значений символам в наборе и использования этих целых значений для кодирования символов строки (см. 7.17.6).

8.3 Таблица алгоритмов кодирования

8.3.1 Каждый документ быстрого инфо-набора имеет связанную с ним таблицу алгоритмов кодирования. Таблица алгоритмов кодирования содержит определения алгоритмов кодирования, на которые можно ссылаться через индекс словарной таблицы.

8.3.2 Каждая запись данной таблицы специфицирует кодирование строки символов с некоторыми заданными характеристиками в строку октетов (см. 7.17.7).

Примечание — Заданные характеристики могут относиться к длине строки, появляющимся в строке символам или к произвольно сложным образцам последовательности символов. В общем случае данный алгоритм кодирования применяют только к ограниченному и определенному подмножеству строк символов по ISO/IEC 10646.

8.3.3 Для алгоритмов кодирования установлены следующие ограничения:

a) алгоритм кодирования, если он не является встроенным, должен иметь связанный с ним URI, по которому на него можно сослаться для добавления в таблицу;

b) алгоритм кодирования должен точно определять, к каким видам строк символов он может быть применен; это определение должно включать в себя алфавит с ограниченной областью распространения (при его наличии), диапазон длин (при его наличии) и любые дополнительные ограничения на длину и содержимое строк символов (например, образец);

c) для любой строки символов, к которой он может быть применен, алгоритм кодирования должен предоставлять обратимое отображение из строки символов в строку октетов.

Примечания

1 Вышеизложенное подразумевает, что не может существовать строка символов S, для которой кодирование из S в E, с последующим декодированием из E в S', приведет к S' ≠ S, даже если различие между S' и S незначительно (например, дополнительный SPACE). С другой стороны, не требуется, чтобы можно было закодировать любую строку символов S, и также не требуется, чтобы кодировки были каноническими.

2 От приложения, создающего документ быстрого инфо-набора из находящихся в памяти данных, таких как числа с плавающей точкой, не требуется создавать лексическое представление этих данных, а затем применять к этому представлению алгоритм кодирования. Вместо этого приложение может создать строку октетов напрямую из этих данных при условии, что полученная строка октетов идентична той, которая могла быть получена путем применения этого алгоритма кодирования к строке символов, представляющей данные в памяти, и к этой строке символов может быть применен данный алгоритм кодирования.

3 Алгоритмы кодирования (отличные от встроенных) могут быть определены в других стандартах или согласованы между создателем и обработчиками документа быстрого инфо-набора.

8.4 Динамические таблицы строк

8.4.1 Каждый документ быстрого инфо-набора имеет восемь связанных с ним динамических таблиц строк. Каждая динамическая таблица строк содержит строки символов, на которые можно ссылаться по индексам словарных таблиц.

8.4.2 В настоящем стандарте все строки символов, которые можно встретить в документе быстрого инфо-набора, классифицированы по следующим восьми категориям, каждая из которых имеет динамическую таблицу строк:

a) PREFIX: данную категорию составляют строки символов, которые являются свойством **[prefix]** информационных элементов **element**, **attribute** или **namespace**;

b) NAMESPACE NAME: данную категорию составляют строки символов, которые являются свойством **[namespace name]** информационных элементов **element**, **attribute** или **namespace**;

c) LOCAL NAME: данную категорию составляют строки символов, которые являются свойством **[local name]** информационных элементов **element** или **attribute**;

d) OTHER NCNAME: данную категорию составляют строки символов, которые являются свойством **[target]** информационного элемента **processing instruction**, свойством **[name]** информационных элементов **unexpanded entity reference**, **unparsed entity** или **notation** или свойством **[notation name]** информационного элемента **unparsed entity**;

e) OTHER URI: данную категорию составляют строки символов, которые являются свойством **[system identifier]** или **[public identifier]** информационных элементов **unexpanded entity reference**, **document type declaration**, **unparsed entity** или **notation**;

f) ATTRIBUTE VALUE: данную категорию составляют строки символов, которые являются свойством **[normalized value]** информационного элемента **attribute**;

g) CONTENT CHARACTER CHUNK: данную категорию составляют строки символов, которые являются свойством **[character code]** блока информационных элементов **character**, являющихся, в свою очередь, последовательными дочерними элементами заданного информационного элемента **element**;

h) OTHER STRING: данную категорию составляют строки символов, которые являются свойством **[version]** информационного элемента **document** или свойством **[content]** информационных элементов **processing instruction** или **comment**.

8.5 Динамические таблицы имен и идентификаторы имен

8.5.1 Каждый документ быстрого инфо-набора имеет две связанные с ним динамические таблицы имен. Каждая динамическая таблица имен содержит идентификаторы имен, на которые можно ссылаться по индексам словарных таблиц и которые используют для идентификации квалифицированных имен. Квалифицированное имя может иметь или не иметь префикс (а также может иметь или не иметь имя пространства имен).

8.5.2 Идентификаторы имен представляют собой набор максимум из трех индексов:

a) (опционально) индекса строки в таблице PREFIX,

b) (опционально) индекса строки в таблице NAMESPACE NAME и

c) индекса строки в таблице LOCAL NAME.

Первый индекс словарной таблицы не должен присутствовать, если отсутствует второй.

8.5.3 Возможны три случая:

a) все три индекса присутствуют, в этом случае идентификатор имени представляет квалифицированное имя с префиксом;

b) присутствуют только второй и третий индексы, в этом случае идентификатор имени представляет квалифицированное имя без префикса, которое имеет имя пространства имен;

c) присутствует только третий индекс, в этом случае идентификатор имени представляет квалифицированное имя без префикса, которое не имеет имени пространства имен.

8.5.4 В настоящем стандарте все квалифицированные имена, которые можно встретить в документе быстрого инфо-набора, классифицированы по двум следующим категориям, каждая из которых имеет динамическую таблицу имен:

a) ELEMENT NAME: данную категорию составляют идентификаторы имен, представляющие квалифицированное имя информационного элемента **element**;

b) ATTRIBUTE NAME: данную категорию составляют идентификаторы имен, представляющие квалифицированное имя информационного элемента **attribute**.

8.5.5 Квалифицированное имя, представленное данным идентификатором имени, нужно определять следующим образом (при заданной динамической таблице строк):

a) первый индекс словарной таблицы (при его наличии) должен интерпретироваться как индекс словарной таблицы строки символов в таблице PREFIX,

b) второй индекс словарной таблицы (при его наличии) должен интерпретироваться как индекс словарной таблицы строки символов в таблице NAMESPACE NAME и

c) третье целое значение должно интерпретироваться как индекс словарной таблицы строки символов в таблице LOCAL NAME.

9 Встроенные алфавиты с ограниченной областью распространения

9.1 «Цифровой» алфавит с ограниченной областью распространения

9.1.1 Данный алфавит с ограниченной областью распространения имеет индекс словарной таблицы 1 и состоит из следующих пятнадцати символов по ISO/IEC 10646 (в указанном порядке):

- DIGIT ZERO — DIGIT NINE
- HYPHEN-MINUS
- PLUS SIGN
- FULL STOP
- LATIN SMALL LETTER E
- SPACE

9.1.2 Данный алфавит с ограниченной областью распространения пригоден для кодирования строк символов, представляющих несколько типов чисел, включая числа с плавающей точкой в научной нотации. Одна строка символов может содержать несколько чисел, разделенных пробелами.

9.2 Алфавит с ограниченной областью распространения «дата и время»

9.2.1 Данный алфавит с ограниченной областью распространения имеет индекс словарной таблицы 2 и состоит из следующих пятнадцати символов по ISO/IEC 10646 (в указанном порядке):

- DIGIT ZERO — DIGIT NINE
- HYPHEN-MINUS
- COLON
- LATIN CAPITAL LETTER T
- LATIN CAPITAL LETTER Z
- SPACE

9.2.2 Данный алфавит с ограниченной областью распространения пригоден для кодирования строк символов, представляющих наиболее общие выражения даты и времени по ISO 8601. Одна строка символов может содержать несколько таких выражений, разделенных пробелами.

10 Встроенные алгоритмы кодирования

10.1 Общие положения

10.1.1 В данном разделе определены встроенные алгоритмы кодирования. Встроенные алгоритмы кодирования не имеют связанных с ними URI.

Примечание — URI необходимы для алгоритмов кодирования, которые должны быть явным образом идентифицированы в исходном словаре, а встроенные алгоритмы кодирования всегда неявно добавляют в таблицу алгоритмов кодирования, и следовательно, они не нуждаются в URI.

10.1.2 В данном разделе термин «слово» обозначает любую группу последовательных символов в данной строке символов, которая:

a) не содержит SPACE и

b) находится в начале строки символов или предваряется SPACE либо

c) находится в конце строки символов или за ней следует SPACE.

Примечание — Термин «слово» не ограничивается алфавитными символами.

10.2 Алгоритм кодирования «hexadecimal»

10.2.1 Данный алгоритм кодирования имеет индекс словарной таблицы 1 и может быть применен только к строкам символов, состоящим из следующих шестнадцати символов по ISO/IEC 10646:

- DIGIT ZERO — DIGIT NINE
- LATIN CAPITAL LETTER A — LATIN CAPITAL LETTER F

и содержащим четное число символов (включая ноль).

Примечание — Использование встроенных пробельных символов XML не допускается.

10.2.2 Строка символов должна интерпретироваться как шестнадцатеричное кодирование строки октетов с первым символом строки, соответствующим самому старшему полубайту первого октета, и т. д.

10.3 Алгоритм кодирования «base64»

10.3.1 Данный алгоритм кодирования имеет индекс словарной таблицы 2 и может быть применен только к строкам символов, которые:

a) состоят полностью из символов LATIN CAPITAL LETTER A — LATIN CAPITAL LETTER Z, LATIN SMALL LETTER A — LATIN SMALL LETTER Z, DIGIT ZERO — DIGIT NINE, PLUS SIGN, SOLIDUS и EQUALS SIGN и

Примечание — Присутствие в строке символов пробельных символов XML не допускается.

b) представляют собой действительный экземпляр кодирования передачи содержимого (Content Transfer Encoding) по IETF RFC 2045, п. 6.8 или становятся действительным экземпляром этого кодирования после вставки пробельных символов XML там, где требуется IETF RFC 2045.

10.3.2 Строка символов должна интерпретироваться как кодирование Base64 (по IETF RFC 2045) строки октетов (при условии, что пробельные символы XML присутствуют там, где требуется IETF RFC 2045). Полученная строка октетов является строкой октетов, определенной кодированием Base64.

10.3.3 Данный алгоритм кодирования пригоден для кодирования строк символов, которые не содержат пробельных символов XML и являются строками Base64 (произвольной длины) или могут ими стать, или после добавления пробельных символов XML.

10.4 Алгоритм кодирования «short»

10.4.1 Данный алгоритм кодирования имеет индекс словарной таблицы 3 и может быть применен только к строкам символов, которые удовлетворяют всем следующим условиям:

a) строка символов полностью состоит из символов DIGIT ZERO — DIGIT NINE, HYPHEN-MINUS и SPACE;

b) первый и последний символы не являются символами SPACE, и в строке нет пары смежных символов SPACE;

c) строка символов содержит по крайней мере одно слово (см. 10.1.2);

d) каждый присутствующий символ HYPHEN-MINUS является первым символом слова;

e) за каждым из символов HYPHEN-MINUS следует по крайней мере один символ DIGIT ONE — DIGIT NINE;

f) каждый символ DIGIT ZERO является единственным символом в слове или ему предшествует символ DIGIT ONE — DIGIT NINE;

g) каждое слово в строке символов, интерпретированное как строка цифровых символов, представляющая целое десятичное значение со знаком, дает значение в диапазоне от минус 32768 до плюс 32767.

10.4.2 Каждое слово (см. 10.1.2) в строке символов должно быть интерпретировано как строка цифровых символов, представляющая целое десятичное значение со знаком, и представлено как 16-битное целое значение с дополнением до двух.

10.4.3 Каждая группа из 8 битов в 16-битном целом с дополнением до двух для слова дает октет строки октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов становится самым старшим битом соответствующего октета. Если в строке символов имеется несколько слов, то они должны быть закодированы по порядку, а октеты, задаваемые 16-битными целыми значениями с дополнением до двух, должны быть сцеплены в этом порядке.

10.4.4 Данный алгоритм кодирования пригоден для кодирования строк символов, представляющих целые значения в диапазоне от минус 32768 до плюс 32767 (представляемые как 16-битное целое значение с дополнением до двух), или их списков.

10.5 Алгоритм кодирования «int»

10.5.1 Данный алгоритм кодирования имеет индекс словарной таблицы 4 и может быть применен только к строкам символов, которые удовлетворяют всем следующим условиям:

a) строка символов удовлетворяет условиям 10.4.1, перечисления a) — f);

b) каждое слово в строке символов, интерпретированное как строка цифровых символов, представляющая целое десятичное значение со знаком, дает значение в диапазоне от минус 2147483648 до плюс 2147483647.

10.5.2 Каждое слово (см. 10.1.2) в строке символов должно быть интерпретировано как строка цифровых символов, представляющая целое десятичное значение со знаком, и представлено как 32-битное целое значение с дополнением до двух.

10.5.3 Каждая группа из 8 битов в 32-битном целом значении с дополнением до двух для слова дает октет строки октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов становится самым старшим битом соответствующего октета. Если в строке символов имеется несколько слов, то они должны быть закодированы по порядку, а октеты, даваемые 32-битными целыми значениями с дополнением до двух, должны быть сцеплены в этом порядке.

10.5.4 Данный алгоритм кодирования применим для кодирования строк символов, представляющих целые значения в диапазоне от минус 2147483648 до плюс 2147483647 (представляемые как 32-битное целое значение с дополнением до двух), или их списков.

10.6 Алгоритм кодирования «long»

10.6.1 Данный алгоритм кодирования имеет индекс словарной таблицы 5. Он может быть применен только к строкам символов, которые удовлетворяют всем следующим условиям:

a) строка символов удовлетворяет условиям 10.4.1, перечисления a) — f);

b) каждое слово в строке символов, интерпретированное как строка цифровых символов, представляющая целое десятичное значение со знаком, дает значение в диапазоне от минус 9223372036854775808 до плюс 9223372036854775807.

10.6.2 Каждое слово (см. 10.1.2) в строке символов должно быть интерпретировано как строка цифровых символов, представляющая целое десятичное значение со знаком, и представлено как 64-битное целое значение с дополнением до двух.

10.6.3 Каждая группа из 8 битов в 64-битном целом с дополнением до двух для слова дает октет строки октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов становится самым старшим битом соответствующего октета. Если в строке символов имеется несколько слов, то они должны быть закодированы по порядку, а октеты, даваемые 64-битными целыми с дополнением до двух, должны быть сцеплены в этом порядке.

10.6.4 Данный алгоритм кодирования применим для кодирования строк символов, представляющих целые значения в диапазоне от минус 9223372036854775808 до плюс 9223372036854775807 (представляемые как 64-битное целое значение с дополнением до двух), или их списков.

10.7 Алгоритм кодирования «boolean»

10.7.1 Данный алгоритм кодирования имеет индекс словарной таблицы 6 и может быть применен только к строкам символов, которые удовлетворяют всем следующим условиям:

a) строка символов полностью состоит из одного или нескольких слов «истина» (true) или «ложь» (false) и символов SPACE;

b) первый и последний символы в строке символов не являются символами SPACE, и в строке нет пары смежных символов SPACE;

c) строка символов содержит по крайней мере одно слово (см. 10.1.2).

10.7.2 Каждое слово «ложь» (false) или «истина» (true) в строке символов должно быть закодировано как один бит (равный нулю или единице соответственно) создаваемой строки октетов, начиная с пятого до восьмого бита первого октета. Последующие биты помещают в последующие октеты, начиная с первого бита каждого октета до восьмого бита этого октета, используя только то количество октетов, которое требуется. Все неиспользованные биты последнего октета должны быть установлены равными нулю.

10.7.3 Первые четыре бита первого октета содержат число неиспользованных битов в последнем октете, закодированное как 4-битное целое значение без знака.

Примечание — Первый октет может быть последним и содержать до трех неиспользованных битов. Если имеется несколько октетов, то последний октет может содержать до семи неиспользованных битов.

10.8 Алгоритм кодирования «float»

10.8.1 Данный алгоритм кодирования имеет индекс словарной таблицы 7 и может быть применен только к строкам символов, которые удовлетворяют всем следующим условиям:

a) строка символов полностью состоит из символов DIGIT ZERO — DIGIT NINE, HYPHEN-MINUS, FULL STOP, LATIN CAPITAL LETTER E и SPACE.

Примечание — Использование символа LATIN SMALL LETTER E не допускается, так как в этом случае кодирование не будет обратимым;

b) первый и последний символы в строке символов не являются символами SPACE, и в строке нет пары смежных символов SPACE;

c) строка символов содержит хотя бы одно слово (см. 10.1.2);

d) каждое слово в строке символов соответствует каноническому лексическому представлению числа с плавающей точкой по схеме W3C XML, часть 2, п. 3.2.4;

e) каждое слово (см. 10.1.2) в строке символов, интерпретированное как строка символов, представляющая десятичное число с плавающей точкой, дает значение, которое может быть представлено как 32-битное число с плавающей точкой по IEEE 754.

10.8.2 Каждое слово (см. 10.1.2) в строке символов должно быть интерпретировано как строка символов, представляющая десятичное число с плавающей точкой, и представлено как 32-битное число с плавающей точкой по IEEE 754.

10.8.3 Каждая группа из 8 битов в 32-битном числе с плавающей точкой по IEEE 754 для слова дает октет в строке октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов становится самым старшим битом соответствующего октета. Если в строке символов есть несколько слов, то они должны быть закодированы по порядку, а октеты, создаваемые 32-битными числами с плавающей точкой по IEEE 754, должны быть сцеплены в этом порядке.

10.8.4 Данный алгоритм кодирования пригоден для кодирования строк символов, представляющих числа с плавающей точкой (представляемые как 32-битные числа с плавающей точкой по IEEE 754) или списки таких чисел.

10.9 Алгоритм кодирования «double»

10.9.1 Данный алгоритм кодирования имеет индекс словарной таблицы 8 и может быть применен только к строкам символов, которые удовлетворяют всем следующим условиям:

a) строка символов удовлетворяет условиям 10.8.1, перечисления a) — c);

b) каждое слово в строке символов соответствует каноническому лексическому представлению числа с плавающей точкой двойной точности по схеме W3C XML, часть 2, п. 3.2.5;

c) каждое слово (см. 10.1.2) в строке символов, интерпретированное как строка символов, представляющая десятичное число с плавающей точкой двойной точности, дает значение, которое может быть представлено как 64-битное число с плавающей точкой двойной точности по IEEE 754.

10.9.2 Каждое слово (см. 10.1.2) в строке символов должно быть интерпретировано как строка символов, представляющая десятичное число с плавающей точкой двойной точности, и должно быть представлено как 64-битное число с плавающей точкой двойной точности по IEEE 754.

10.9.3 Каждая группа из 8 битов в 64-битном числе с плавающей точкой двойной точности по IEEE 754 для слова дает октет в строке октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов становится самым старшим битом соответствующего октета. Если в строке символов есть несколько слов, то они должны быть закодированы по порядку, а октеты, создаваемые 64-битными числами с плавающей точкой двойной точности по IEEE 754, должны быть сцеплены в этом порядке.

10.9.4 Данный алгоритм кодирования пригоден для кодирования строк символов, представляющих числа с плавающей точкой (представляемые как 64-битные числа с плавающей точкой двойной точности по IEEE 754) или списки таких чисел.

10.10 Алгоритм кодирования «uuid»

10.10.1 Данный алгоритм кодирования имеет индекс словарной таблицы 8 и может быть применен только к строкам символов, которые удовлетворяют всем следующим условиям:

a) строка символов полностью состоит из символов DIGIT ZERO — DIGIT NINE, LATIN SMALL LETTER A — LATIN SMALL LETTER F, HYPHEN-MINUS и SPACE;

b) первый и последний символы в строке символов не являются символами SPACE, и в строке нет пары смежных символов SPACE;

c) строка символов содержит хотя бы одно слово (см. 10.1.2);

d) каждое слово содержит ровно 36 символов;

e) в каждом слове есть ровно четыре символа HYPHEN-MINUS, занимающих позиции 9, 14, 19 и 24 (считая от единицы).

10.10.2 Каждое слово в строке символов должно интерпретироваться как шестнадцатеричное представление UUID (по ISO/IEC 9834-8, п. 6.4) и быть представлено как 16-битное целое значение без

знака, как определено в ISO/IEC 9834-8, п. 6.3. Если в строке символов несколько слов, то соответствующие 16-октетные целые значения без знака должны быть сцеплены.

10.10.3 Данный алгоритм кодирования пригоден для кодирования строк символов, представляющих UUID или список UUID.

10.11 Алгоритм кодирования «cdata»

10.11.1 Данный алгоритм кодирования имеет индекс словарной таблицы 8 и может быть применен к любой строке символов.

10.11.2 Полученная строка октетов должна быть кодированием UTF-8 (по ISO/IEC 10646) строки символов.

10.11.3 Данный алгоритм следует использовать только с инфо-наборами XML, созданными в результате синтаксического анализа документа XML, когда дополнительная информация идентифицирует, что строка символов соответствует всему разделу CDATA (по W3C XML 1.0 и W3C XML 1.1). Если данный алгоритм кодирования используют в документе быстрого инфо-набора, то ко всем строкам символов, которые соответствуют разделам CDATA, следует применять этот алгоритм кодирования.

11 Ограничения на поддерживаемые инфо-наборы XML и некоторые упрощения

11.1 Настоящий стандарт поддерживает большинство инфо-наборов XML, которые могут встретиться на практике, однако не поддерживает некоторые инфо-наборы XML, которые возможны только теоретически.

11.2 В настоящем разделе термин «XML-самосогласованный» используют в следующем значении: набор свойств одного или нескольких информационных элементов является XML-самосогласованным, если этот набор свойств может быть получен в результате синтаксического анализа подходящего корректно сформированного в отношении пространств имен документа XML.

11.3 Поддерживаемые инфо-наборы XML удовлетворяют следующим условиям:

a) свойство **[all declarations processed]** информационного элемента **document** имеет значение **true** (истина);

b) свойство **[in-scope namespaces]** каждого информационного элемента **element** и свойства **[namespace attributes]** всех информационных элементов **element** вместе образуют XML-самосогласованный набор;

c) свойство **[namespace name]** каждого информационного элемента **element**, свойства **[namespace attributes]** всех информационных элементов **element** и свойство **[prefix]** этого информационного элемента **element** вместе образуют XML-самосогласованный набор;

d) свойство **[namespace name]** каждого информационного элемента **attribute**, свойства **[namespace attributes]** всех информационных элементов **element** и свойство **[prefix]** этого информационного элемента **attribute** вместе образуют XML-самосогласованный набор;

e) свойство **[references]** каждого информационного элемента **attribute** и свойство **[normalized value]** информационного элемента **attribute** вместе образуют XML-самосогласованный набор;

f) свойство **[notation]** каждого информационного элемента **processing instruction**, свойство **[target]** информационного элемента **processing instruction** и свойство **[notations]** информационного элемента **document** вместе образуют XML-самосогласованный набор;

g) свойство **[notation]** каждого информационного элемента **unparsed entity**, свойство **[notation name]** информационного элемента **unparsed entity** и свойство **[notations]** информационного элемента **document** вместе образуют XML-самосогласованный набор;

h) свойство **[element content whitespace]** всех информационных элементов **character**, которые не представляют пробельные символы, имеет значение **false** (ложь);

i) свойство **[element content whitespace]** каждого информационного элемента **character** и свойство **[character code]** информационного элемента **character** вместе образуют XML-самосогласованный набор;

j) свойство **[normalized value]** всех информационных элементов **attribute** и свойство **[content]** всех информационных элементов **comment** и **processing instruction** содержат не более 2^{32} символов.

11.4 Следующие свойства информационных элементов инфо-набора XML не включены в типы ASN.1, представляющие эти информационные элементы:

a) свойства **[document element]**, **[base URI]** и **[all declarations processed]** информационного элемента **document** (см. 7.2.30, 7.2.31 и 7.2.32);

b) свойства **[in-scope namespaces]**, **[base URI]** и **[parent]** информационного элемента **element** (см. 7.3.8, 7.3.9 и 7.3.10);

c) свойства **[specified]**, **[attribute type]**, **[references]** и **[owner element]** информационного элемента **attribute** (см. 7.4.7, 7.4.8, 7.4.9 и 7.4.10);

d) свойства **[notation]** и **[parent]** информационного элемента **processing instruction** (см. 7.5.7 и 7.5.8);

e) свойства **[declaration base URI]** и **[parent]** информационного элемента **unexpanded entity reference** (см. 7.6.7 и 7.6.8);

f) свойство **[element content whitespace]** информационного элемента **character** (см. 7.7.7);

g) свойство **[parent]** информационного элемента **character** (см. 7.7.8);

h) свойство **[parent]** информационного элемента **comment** (см. 7.8.6);

i) свойство **[parent]** информационного элемента **document type declaration** (см. 7.9.7);

j) свойства **[declaration base URI]** и **[notation]** информационного элемента **unparsed entity** (см. 7.10.8 и 7.10.9);

k) свойство **[declaration base URI]** информационного элемента **notation** (см. 7.11.7).

12 Битовое кодирование типа `Document`

12.1 В данном разделе определены специальные кодирования типа `Document` для создания документа быстрого инфо-набора.

Примечание — Эти специальные кодирования разработаны для оптимизации скорости обработки и компактности, которые считаются критическими во многих предполагаемых случаях использования настоящего стандарта.

12.2 Кодирования определены в терминах действий, которые должны быть выполнены кодировщиком и привести к присоединению битов к потоку битов. Исходный поток битов либо пуст, либо состоит из декларации XML (см. 12.3).

12.3 Декларация XML (по W3C XML 1.1, п. 2.8) может быть (по выбору создателя документа быстрого инфо-набора) включена в начало потока битов. Декларация XML (при ее наличии) должна быть одной из следующих строк символов, закодированной в UTF-8:

1) **<?xml encoding='finf'?>**
2) **<?xml encoding='finf' standalone='yes'?>**
3) **<?xml encoding='finf' standalone='no'?>**
4) **<?xml version='1.0' encoding='finf'?>**
5) **<?xml version='1.0' encoding='finf' standalone='yes'?>**
6) **<?xml version='1.0' encoding='finf' standalone='no'?>**
7) **<?xml version='1.1' encoding='finf'?>**
8) **<?xml version='1.1' encoding='finf' standalone='yes'?>**
9) **<?xml version='1.1' encoding='finf' standalone='no'?>**

12.4 Номер версии (при его наличии) в декларации XML должен быть установлен равным соответствующему свойству **[version]** информационного элемента **document**. Декларация XML не должна содержать номер версии, если свойство **[version]** не имеет значения.

12.5 Декларация отдельного расположения (при ее наличии) в декларации XML должна быть установлена равной соответствующему свойству **[standalone]** информационного элемента **document**. Декларация XML не должна содержать декларации отдельного расположения, если свойство **[standalone]** не имеет значения.

12.6 В конце потока битов должны быть добавлены шестнадцать битов '1110000000000000'.

Примечание — Указанные биты будут находиться либо в начале документа быстрого инфо-набора, либо следом за декларацией XML. При отсутствии декларации XML синтаксический анализатор может отличить, просматривая первые 16 битов кодирования, потенциальный документ быстрого инфо-набора от любого другого корректно сформированного документа W3C XML 1.0 или W3C XML 1.1, так как эти 16 битов никогда не встречаются в начале корректно сформированного документа XML.

12.7 Далее к потоку битов должно быть добавлено битовое поле из 16 битов, содержащее номер версии настоящего стандарта (см. 12.9), закодированный как 16-битное целое значение без знака.

12.8 Далее к потоку битов должен быть добавлен бит '0' (забивка).

Примечание — Это делается для обеспечения выравнивания байтов в последующих частях кодирования.

12.9 Номер версии данной редакции настоящего стандарта равен 1.

Примечание — Предполагается, что в последующих редакциях настоящего стандарта номер версии будет увеличен во избежание проблем взаимодействия между новой и предыдущими редакциями.

12.10 К концу потока битов должно быть добавлено кодирование ECN (по ISO/IEC 8825-3) абстрактного значения типа **`Document`**, определенное модулем связи кодирования в А.2.

Примечание — В Приложении С приведено неформальное описание кодирований, определенных в А.2.

12.11 Если кодирование абстрактного значения типа **`Document`** не заканчивается на последнем бите октета, то к потоку битов должны быть добавлены четыре бита '0000' (забивка), завершающие последний октет.

12.12 После выполнения всех указанных выше шагов содержимое потока битов является документом быстрого инфо-набора.

Приложение А
(обязательное)

Модуль ASN.1 и модули ECN для документов быстрого инфо-набора

А.1 Определение модуля ASN.1

```
FastInfoset {joint-iso-itu-t(2) asn1(1) generic-applications(10)
fast-infoset(0) modules(0) fast-infoset(0)}

DEFINITIONS AUTOMATIC TAGS ::= BEGIN

finf-doc-opt-decl OBJECT IDENTIFIER ::= {joint-iso-itu-t(2) asn1(1)
     generic-applications(10) fast-infoset(0) encodings(1)
     optional-xml-declaration(0)}
finf-doc-no-decl OBJECT IDENTIFIER ::= {joint-iso-itu-t(2) asn1(1)
     generic-applications(10) fast-infoset(0) encodings(1)
     no-xml-declaration(1)}

Document ::= SEQUENCE {
     additional-data SEQUENCE (SIZE(1..one-meg)) OF
          additional-datum SEQUENCE {
               id URI,
               data NonEmptyOctetString } OPTIONAL,
          initial-vocabulary SEQUENCE {
               external-vocabulary URI OPTIONAL,
               restricted-alphabets SEQUENCE (SIZE(1..256)) OF
                    NonEmptyOctetString OPTIONAL,
               encoding-algorithms SEQUENCE (SIZE(1..256)) OF
                    NonEmptyOctetString OPTIONAL,
               prefixes SEQUENCE (SIZE(1..one-meg)) OF
                    NonEmptyOctetString OPTIONAL,
               namespace-names SEQUENCE (SIZE(1..one-meg)) OF
                    NonEmptyOctetString OPTIONAL,
               local-names SEQUENCE (SIZE(1..one-meg)) OF
                    NonEmptyOctetString OPTIONAL,
               other-ncnames SEQUENCE (SIZE(1..one-meg)) OF
                    NonEmptyOctetString OPTIONAL,
               other-uris SEQUENCE (SIZE(1..one-meg)) OF
                    NonEmptyOctetString OPTIONAL,
               attribute-values SEQUENCE (SIZE(1..one-meg)) OF
                    EncodedCharacterString OPTIONAL,
               content-character-chunks SEQUENCE (SIZE(1..one-meg)) OF
                    EncodedCharacterString OPTIONAL,
               other-strings SEQUENCE (SIZE(1..one-meg)) OF
                    EncodedCharacterString OPTIONAL,
               element-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
                    NameSurrogate OPTIONAL,
              attribute-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
                    NameSurrogate OPTIONAL }
                     (CONSTRAINED BY {
                          -- Если присутствует компонент
                          -- initial-vocabulary, то должен
                          -- присутствовать хотя бы один
                          -- из его компонентов -- }) OPTIONAL,
          notations SEQUENCE (SIZE(1..MAX)) OF
```

```
            Notation OPTIONAL,
  unparsed-entities SEQUENCE (SIZE(1..MAX)) OF
            UnparsedEntity OPTIONAL,
  character-encoding-scheme NonEmptyOctetString OPTIONAL,
  standalone BOOLEAN OPTIONAL,
  version NonIdentifyingStringOrIndex OPTIONAL
            -- Категория OTHER STRING --,
children SEQUENCE (SIZE(0..MAX)) OF
    CHOICE {
        element Element,
        processing-instruction ProcessingInstruction,
        comment Comment,
        document-type-declaration DocumentTypeDeclaration }}
one-meg INTEGER ::= 1048576 -- Два в степени 20
four-gig INTEGER ::= 4294967296 -- Два в степени 32
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
URI ::= NonEmptyOctetString
Element ::= SEQUENCE {
   namespace-attributes  SEQUENCE (SIZE(1..MAX)) OF
        NamespaceAttribute OPTIONAL,
   qualified-name QualifiedNameOrIndex
                -- Категория ELEMENT NAME --,
   attributes SEQUENCE (SIZE(1..MAX)) OF
        Attribute OPTIONAL,
   children SEQUENCE (SIZE(0..MAX)) OF
        CHOICE {
            element Element,
            processing-instruction ProcessingInstruction,
            unexpanded-entity-reference
                        UnexpandedEntityReference,
            character-chunk CharacterChunk,
            comment Comment }}
Attribute ::= SEQUENCE {
    qualified-name QualifiedNameOrIndex
            -- Категория ATTRIBUTE NAME --,
    normalized-value NonIdentifyingStringOrIndex
            -- Категория ATTRIBUTE VALUE -- }
ProcessingInstruction ::= SEQUENCE {
    target IdentifyingStringOrIndex
            -- Категория OTHER NCNAME --,
    content NonIdentifyingStringOrIndex
            -- Категория OTHER STRING -- }
UnexpandedEntityReference ::= SEQUENCE {
    name IdentifyingStringOrIndex
            -- Категория OTHER NCNAME --,
    system-identifier IdentifyingStringOrIndex OPTIONAL
            -- Категория OTHER URI --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
            -- Категория OTHER URI -- }
CharacterChunk ::= SEQUENCE {
    character-codes NonIdentifyingStringOrIndex
            -- Категория CONTENT CHARACTER CHUNK -- }
Comment ::= SEQUENCE {
    content NonIdentifyingStringOrIndex
            -- Категория OTHER STRING -- }
```

```
DocumentTypeDeclaration ::= SEQUENCE {
    system-identifier IdentifyingStringOrIndex OPTIONAL
            -- Категория OTHER URI --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
            -- Категория OTHER URI --,
    children SEQUENCE (SIZE(0..MAX)) OF
        ProcessingInstruction }
UnparsedEntity ::= SEQUENCE {
    name IdentifyingStringOrIndex
            -- Категория OTHER NCNAME --,
    system-identifier IdentifyingStringOrIndex
            -- Категория OTHER URI --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
            -- Категория OTHER URI --,
    notation-name IdentifyingStringOrIndex
            -- Категория OTHER NCNAME -- }
Notation ::= SEQUENCE {
    name IdentifyingStringOrIndex
            -- Категория OTHER NCNAME --,
    system-identifier IdentifyingStringOrIndex OPTIONAL
            -- Категория OTHER URI --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
            -- Категория OTHER URI -- }
NamespaceAttribute ::= SEQUENCE {
    prefix IdentifyingStringOrIndex OPTIONAL
            -- Категория PREFIX --,
    namespace-name IdentifyingStringOrIndex OPTIONAL
            -- Категория NAMESPACE NAME -- }
IdentifyingStringOrIndex ::= CHOICE {
    literal-character-string NonEmptyOctetString,
    string-index INTEGER (1..one-meg) }
NonIdentifyingStringOrIndex ::= CHOICE {
    literal-character-string SEQUENCE {
        add-to-table BOOLEAN,
        character-string EncodedCharacterString },
    string-index INTEGER (0..one-meg) }
NameSurrogate ::= SEQUENCE {
    prefix-string-index INTEGER(1..one-meg) OPTIONAL,
    namespace-name-string-index INTEGER(1..one-meg) OPTIONAL,
    local-name-string-index INTEGER(1..one-meg) }
     (CONSTRAINED BY {-- prefix-string-index должен
            -- присутствовать, только если
            -- присутствует namespace-name-string-index
            -- })
QualifiedNameOrIndex ::= CHOICE {
    literal-qualified-name SEQUENCE {
        prefix IdentifyingStringOrIndex OPTIONAL
                -- Категория PREFIX --,
        namespace-name IdentifyingStringOrIndex OPTIONAL
                -- Категория NAMESPACE NAME --,
        local-name IdentifyingStringOrIndex
                -- Категория LOCAL NAME -- },
    name-surrogate-index INTEGER (1..one-meg) }
EncodedCharacterString ::= SEQUENCE {
    encoding-format CHOICE {
```

```
                utf-8 NULL,
                utf-16 NULL,
                restricted-alphabet INTEGER(1..256),
                encoding-algorithm INTEGER(1..256) },
            octets NonEmptyOctetString }
    END
```

A.2 Определения модуля ECN

```
FastInfosetEDM {joint-iso-itu-t(2) asn1(1) generic-applications(10)
fast-infoset(0) modules(0) fast-infoset-edm(1)}
   ENCODING-DEFINITIONS ::= BEGIN
   EXPORTS FastInfosetEncodingSet;
   RENAMES
            #INTEGER AS #PositiveOrNonNegativeInteger
            IN #IdentifyingStringOrIndex.string-index,
               #NonIdentifyingStringOrIndex.string-index,
               #QualifiedNameOrIndex.name-surrogate-index,
               #NameSurrogate.namespace-name-string-index,
               #NameSurrogate.prefix-string-index,
               #NameSurrogate.local-name-string-index
            FROM FastInfoset;
   /* RENAMES автоматически импортирует:
   #Document, #NonEmptyOctetString, #NameSurrogate,
   #ProcessingInstruction,#UnexpandedEntityReference,
   #Comment, #DocumentTypeDeclaration, #UnparsedEntity,
   #Notation, #Element, #Attribute, #CharacterChunk,
   #NamespaceAttribute, #IdentifyingStringOrIndex,
   #NonIdentifyingStringOrIndex, #QualifiedNameOrIndex,
   #EncodedCharacterString FROM FastInfoset;
   */
-- Полезные классы кодирования
#PositiveOrNonNegativeInteger ::= #INTEGER
#NonEmptySequenceOfLength ::= #INT(1..1048576)
#NonEmptyOctetStringLength ::= #INT(1..4294967296)
#TwoAlternativeDiscriminant ::= #INT(0..1)
#ThreeAlternativeDiscriminant ::= #INT(0..2)
#FourAlternativeDiscriminant ::= #INT(0..3)
#FiveAlternativeDiscriminant ::= #INT(0..4)
-- Используют при кодировании длины SEQUENCE OF (см. С.21)
#NonEmptySequenceOfLengthAlternatives1 ::= #ALTERNATIVES {
    small #INT(1..128),
    large #INT(129..1048576) }
-- Используют при кодировании длины NonEmptyOctetString (см. С.22)
#NonEmptyOctetStringLengthAlternatives2 ::= #ALTERNATIVES {
    small #INT(1..64),
    medium #INT(65..320),
    large #INT(321..4294967296) }
-- Используют при кодировании длины NonEmptyOctetString (см. С.23)
#NonEmptyOctetStringLengthAlternatives5 ::= #ALTERNATIVES {
    small #INT(1..8),
    medium #INT(9..264),
    large #INT(265..4294967296) }
-- Используют при кодировании длины NonEmptyOctetString (см. С.24)
#NonEmptyOctetStringLengthAlternatives7 ::= #ALTERNATIVES {
```

```
    small #INT(1..2),
    medium #INT(3..258),
    large #INT(259..4294967296) }
-- Используют при кодировании положительного целого значения (см. С.25)
#PositiveIntegerAlternatives2 ::= #ALTERNATIVES {
    small #INT(1..64),
    medium #INT(65..8256),
    large #INT(8257..1048576) }
-- Используют при кодировании положительного целого значения (см. С.27)
#PositiveIntegerAlternatives3 ::= #ALTERNATIVES {
    small #INT(1..32),
    medium #INT(33..2080),
    medium-large #INT(2081..526368),
    large #INT(526369..1048576) }
-- Используют при кодировании положительного целого значения (см. С.28)
#PositiveIntegerAlternatives4 ::= #ALTERNATIVES {
    small #INT(1..16),
    medium #INT(17..1040),
    medium-large #INT(1041..263184),
    large #INT(263185..1048576) }
-- Используют при кодировании неотрицательного целого значения (см. С.26)
#NonNegativeIntegerAlternatives2 ::= #ALTERNATIVES {
    zero #INT(0),
    small #INT(1..64),
    medium #INT(65..8256),
    large #INT(8257..1048576) }
-- Во многих случаях используют для вставки начальной забивки
#PrecededByPrepadding{<#C>} ::= #CONCATENATION {
    prepadding #PAD,
    original #C }
-- Используют для вставки двухвариантного дискриминанта
-- перед кодированием
#PrecededByTwoAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
    discriminant #TwoAlternativeDiscriminant,
    original #C }
-- Используют для вставки трехвариантного дискриминанта
-- перед кодированием
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
    discriminant #ThreeAlternativeDiscriminant,
    original #C }
-- Используют для вставки четырехвариантного дискриминанта
-- перед кодированием
#PrecededByFourAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
    prepadding #PAD,
    discriminant #FourAlternativeDiscriminant,
    original #C }
-- Используют для вставки пятивариантного дискриминанта
-- перед кодированием
#PrecededByFiveAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
    prepadding #PAD,
    discriminant #FiveAlternativeDiscriminant,
    original #C }
-- Используют для вставки поля длины перед кодированием SEQUENCE OF
#PrecededByNonEmptySequenceOfLength{<#C>} ::= #CONCATENATION {
    length #NonEmptySequenceOfLength,
```

```
        original #C }
    -- Используют для вставки поля длины перед
    -- кодированием NonEmptyOctetString
    #PrecededByNonEmptyOctetStringLength{<#C>} ::= #CONCATENATION {
        length #NonEmptyOctetStringLength,
        original #C }
    -- Кодируют элемент компонента notations типа Document (см. C.2.6.1)
    eNotationDriver1 #Notation ::= {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNotationPrepaddingAdder1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют элемент компонента unparsed-entities типа Document (см. C.2.7.1)
    eUnparsedEntityDriver1 #UnparsedEntity ::= {
        ENCODE STRUCTURE {
            STRUCTURED WITH eUnparsedEntityPrepaddingAdder1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют элемент компонента namespace-attributes типа Element (см. C.3.4.2)
    eNamespaceAttributeDriver1 #NamespaceAttribute ::= {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNamespaceAttributePrepaddingAdder1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют элемент компонента attributes типа Element (см. C.3.6.1)
    eAttributeDriver1 #Attribute ::= {
        ENCODE STRUCTURE {
            STRUCTURED WITH eAttributePrepaddingAdder1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют элемент компонентов attribute-values,
    -- content-character-chunks и other-strings типа Document (см. C.2.5.4)
    eEncodedCharacterStringDriver1 #EncodedCharacterString ::= {
        ENCODE STRUCTURE {
            STRUCTURED WITH eEncodedCharacterStringPrepaddingAdder1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют элемент компонентов element-name-surrogates и
    -- attribute-name-surrogates типа Document (см. C.2.5.5)
    eNameSurrogateDriver1 #NameSurrogate ::= {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNameSurrogatePrepaddingAdder1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют компонент initial-vocabulary типа Document (см. C.2.5)
    eInitialVocabularyPrepaddingAdder1 #SEQUENCE ::= {
        REPLACE STRUCTURE WITH #PrecededByPrepadding
        ENCODED BY eInitialVocabularyWithPrepadding1 }
    -- Вставляют начальную забивку перед кодированием элемента
    -- компонента notations типа Document (см. C.2.6.1)
    eNotationPrepaddingAdder1 #SEQUENCE ::= {
        REPLACE STRUCTURE WITH #PrecededByPrepadding
        ENCODED BY eNotationWithPrepadding1 }
    -- Вставляют начальную забивку перед кодированием элемента
    -- компонента unparsed-entities типа Document (см. C.2.7.1)
    eUnparsedEntityPrepaddingAdder1 #SEQUENCE ::= {
        REPLACE STRUCTURE WITH #PrecededByPrepadding
        ENCODED BY eUnparsedEntityWithPrepadding1 }
    -- Вставляют начальную забивку перед кодированием элемента
    -- компонента standalone типа Document (см. C.2.9)
```

```
eStandalonePrepaddingAdder1 #BOOL ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eStandaloneWithPrepadding1 }
-- Вставляют начальную забивку перед кодированием элемента
-- компонента children типа DocumentTypeDeclaration (см. C.9.6)
eDocTypeDeclChildPrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eDocTypeDeclChildWithPrepadding1 }
-- Вставляют начальную забивку перед кодированием элемента
-- компонента namespace-attributes типа Element (см. C.3.4.2)
eNamespaceAttributePrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eNamespaceAttributeWithPrepadding1 }
-- Вставляют начальную забивку перед кодированием элемента
-- компонента attributes типа Element (см. C.3.6.1)
eAttributePrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eAttributeWithPrepadding1 }
-- Вставляют начальную забивку перед кодированием элемента
-- компонента literal-qualified-name типа QualifiedNameOrIndex.
-- Используют, когда кодирование начинается со второго бита
-- октета (см. C.17.3)
eLiteralQualifiedNamePrepaddingAdder2 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eLiteralQualifiedNameWithPrepadding2 }
-- Вставляют начальную забивку перед кодированием элемента
-- компонента literal-qualified-name типа QualifiedNameOrIndex.
-- Используют, когда кодирование начинается с третьего бита
-- октета (см. C.18.3)
eLiteralQualifiedNamePrepaddingAdder3 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eLiteralQualifiedNameWithPrepadding3 }
-- Вставляют начальную забивку перед кодированием элемента
-- компонентов restricted-alphabets, encoding-algorithms,
-- prefixes, namespace-names, local-names, other-ncnames и
-- other-uris типа Document (см. C.2.5.3)
eNonEmptyOctetStringPrepaddingAdder1 #OCTET-STRING ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByPrepadding
        ENCODED BY eNonEmptyOctetStringWithPrepadding1 }}
-- Вставляют начальную забивку перед кодированием элемента
-- компонентов attribute-values, content-character-chunks и
-- other-strings типа Document (см. C.2.5.4)
eEncodedCharacterStringPrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eEncodedCharacterStringWithPrepadding1 }
-- Вставляют начальную забивку перед кодированием элемента
-- компонентов element-name-surrogates и
-- attribute-name-surrogates типа Document (см. C.2.5.5)
eNameSurrogatePrepaddingAdder1 #SEQUENCE ::= {
    REPLACE STRUCTURE WITH #PrecededByPrepadding
    ENCODED BY eNameSurrogateWithPrepadding1 }
-- Вставляют дискриминант перед кодированием элемента компонента
-- children типа Document (см. C.2.11.2 -- C.2.11.5)
```

```
eDocumentChildDiscriminantAdder1or5 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
    ENCODED BY eDocumentChildWithDiscriminant1or5 }
-- Вставляют дискриминант перед кодированием элемента компонента
-- children типа Element (см. C.3.7.2 -- C.3.7.6)
eElementChildDiscriminantAdder1or5 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFiveAlternativeDiscriminant
    ENCODED BY eElementChildWithDiscriminant1or5 }
-- Вставляют дискриминант перед кодированием длины SEQUENCE OF,
-- идентифицируя один из двух способов кодирования длины (см. C.21)
eNonEmptySequenceOfLengthDiscriminantAdder1 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByTwoAlternativeDiscriminant
    ENCODED BY eNonEmptySequenceOfLengthWithDiscriminant1 }
-- Вставляют дискриминант перед кодированием длины
-- NonEmptyOctetString, идентифицируя один из трех способов
-- кодирования длины. Используют, когда кодирование
-- начинается со второго бита октета (см. C.22)
eNonEmptyOctetStringLengthDiscriminantAdder2 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
    ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant2 }
-- Вставляют дискриминант перед кодированием длины
-- NonEmptyOctetString, идентифицируя один из трех способов
-- кодирования длины. Используют, когда кодирование начинается
-- с пятого бита октета (см. C.23)
eNonEmptyOctetStringLengthDiscriminantAdder5 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
    ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant5 }
-- Вставляют дискриминант перед кодированием длины
-- NonEmptyOctetString, идентифицируя один из трех способов
-- кодирования длины. Используют, когда кодирование начинается
-- с седьмого бита октета (см. C.24)
eNonEmptyOctetStringLengthDiscriminantAdder7 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
    ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant7 }
-- Вставляют дискриминант перед кодированием положительного
-- целого значения, идентифицируя один из трех способов его
-- кодирования. Используют, когда кодирование начинается со
-- второго бита октета (см. C.25)
ePositiveIntegerDiscriminantAdder2 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
    ENCODED BY ePositiveIntegerWithDiscriminant2 }
-- Вставляют дискриминант перед кодированием положительного
-- целого значения, идентифицируя один из четырех способов его
-- кодирования. Используют, когда кодирование начинается с
-- третьего бита октета (см. C.27)
ePositiveIntegerDiscriminantAdder3 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
    ENCODED BY ePositiveIntegerWithDiscriminant3 }
-- Вставляют дискриминант перед кодированием положительного
-- целого значения, идентифицируя один из четырех способов его
-- кодирования. Используют, когда кодирование начинается с
-- четвертого бита октета (см. C.28)
ePositiveIntegerDiscriminantAdder4 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
```

```
    ENCODED BY ePositiveIntegerWithDiscriminant4 }
-- Вставляют дискриминант перед кодированием неотрицательного
-- целого значения, идентифицируя один из трех способов его
-- кодирования (см. С.26)
eNonNegativeIntegerDiscriminantAdder2 #CHOICE ::= {
    REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
    ENCODED BY eNonNegativeIntegerWithDiscriminant2 }
-- Устанавливают начальную забивку, которая была добавлена
-- перед компонентом initial-vocabulary типа Document, и
-- кодируют этот компонент (см. С.2.5)
eInitialVocabularyWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 3
            PAD-PATTERN bits:'000'B },
        original {
            ENCODE STRUCTURE {
                restricted-alphabets {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                            eRepetitionWithLengthNonEmptyOctetString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                encoding-algorithms {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                            eRepetitionWithLengthNonEmptyOctetString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                prefixes {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                            eRepetitionWithLengthNonEmptyOctetString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                namespace-names {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                            eRepetitionWithLengthNonEmptyOctetString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                local-names {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                            eRepetitionWithLengthNonEmptyOctetString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                other-ncnames {
                    ENCODE STRUCTURE {
```

```
                        STRUCTURED WITH
                            eRepetitionWithLengthNonEmptyOctetString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                other-uris {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                            eRepetitionWithLengthNonEmptyOctetString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                attribute-values {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                        eRepetitionWithLengthEncodedCharacterString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                content-character-chunks {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                        eRepetitionWithLengthEncodedCharacterString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                other-strings {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                        eRepetitionWithLengthEncodedCharacterString1
    }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                element-name-surrogates {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                            eRepetitionWithLengthNameSurrogate1 }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET,
                attribute-name-surrogates {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                            eRepetitionWithLengthNameSurrogate1 }
                    WITH FastInfosetEncodingSet }
                        OPTIONAL-ENCODING USE-SET }
                WITH FastInfosetEncodingSet }}
            WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена перед
-- каждым элементом компонента notations типа Document, и
-- кодируют этот элемент (см. C.2.6.1)
eNotationWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 6
            PAD-PATTERN bits:'110000'B },
        original eNotation7 }
```

```
    WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена перед
-- каждым элементом компонента unparsed-entities типа Document,
-- и кодируют этот элемент (см. C.2.7.1)
eUnparsedEntityWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 7
            PAD-PATTERN bits:'1101000'B },
        original eUnparsedEntity8 }
    WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена перед
-- компонентом standalone типа Document, и кодируют этот
-- компонент (см. C.2.9)
eStandaloneWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 7
            PAD-PATTERN bits:'0000000'B },
        original USE-SET }
    WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена перед
-- каждым элементом компонента namespace-attributes типа
-- Element, и кодируют этот элемент (см. C.3.4.2)
eNamespaceAttributeWithPrepadding1{<#C>}
                              #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 6
            PAD-PATTERN bits:'110011'B
            EXHIBITS HANDLE «nsa» AT { 0 | 1 | 2 | 3 | 4 | 5}
            AS bits:'110011'B },
        original eNamespaceAttribute7
        STRUCTURED WITH {
            ENCODING-SPACE
            EXHIBITS HANDLE «nsa» AT { 0 | 1 | 2 | 3 | 4 | 5}
            AS bits:'110011'B }}
    WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена перед
-- каждым элементом компонента attributes типа Element, и
-- кодируют этот элемент (см. C.3.6.1)
eAttributeWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 1
            PAD-PATTERN bits:'0'B },
        original eAttribute2 }
    WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена перед
-- каждым элементом компонента children типа
-- DocumentTypeDeclaration, и кодируют этот элемент (см. C.9.6)
eDocTypeDeclChildWithPrepadding1{<#C>}
#PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 8
```

```
            PAD-PATTERN bits:'11100001'B },
        original eProcessingInstruction1 }
    WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена
-- перед каждым элементом компонентов restricted-alphabets,
-- encoding-algorithms, prefixes, namespace-names, local-names,
-- other-ncnames и other-uris типа Document, и кодируют этот
-- элемент (см. C.2.5.3)
eNonEmptyOctetStringWithPrepadding1{<#C>}
#PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 1
            PAD-PATTERN bits:'0'B },
        original USE-SET }
    WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена перед
-- каждым элементом компонентов attribute-values,
-- content-character-chunks и other-strings типа Document, и
-- кодируют этот элемент (см. C.2.5.4)
eEncodedCharacterStringWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 2
            PAD-PATTERN bits:'00'B },
        original USE-SET }
    WITH FastInfosetEncodingSet }
eNameSurrogateWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 6
            PAD-PATTERN bits:'000000'B },
        original eNameSurrogate7 }
    WITH FastInfosetEncodingSet }
-- Устанавливают начальную забивку, которая была добавлена перед
-- компонентом literal-qualified-name типа QualifiedNameOrIndex,
-- и кодируют этот компонент. Используют, когда кодирование
-- начинается со второго бита октета (см. C.17.3)
eLiteralQualifiedNameWithPrepadding2{<#C>} #PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ENCODING-SPACE SIZE 5
            PAD-PATTERN bits:'11110'B },
        original {
            ENCODE STRUCTURE {
                prefix eIdentifyingStringOrIndex1
                        OPTIONAL-ENCODING USE-SET,
                namespace-name eIdentifyingStringOrIndex1
                        OPTIONAL-ENCODING USE-SET,
                local-name eIdentifyingStringOrIndex1 }
            WITH FastInfosetEncodingSet }
        STRUCTURED WITH {
            ENCODING-SPACE
            EXHIBITS HANDLE «qn» AT { 0 | 1 | 2 | 3 }
            AS bits:'1111'B }}
    WITH FastInfosetEncodingSet }
```

```
-- Устанавливает начальную забивку, которая была добавлена перед
-- компонентом literal-qualified-name типа QualifiedNameOrIndex,
-- и кодирует этот компонент. Используют, когда кодирование
-- начинается с третьего бита октета (см. C.18.3)
eLiteralQualifiedNameWithPrepadding3{<#C>}
#PrecededByPrepadding{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
                ENCODING-SPACE SIZE 4
                PAD-PATTERN bits:'1111'B
                EXHIBITS HANDLE «qn» AT { 0 | 1 | 2 | 3 }
                AS bits:'1111'B },
        original {
            ENCODE STRUCTURE {
                prefix eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
                namespace-name eIdentifyingStringOrIndex1
                        OPTIONAL-ENCODING USE-SET,
                local-name eIdentifyingStringOrIndex1 }
            WITH FastInfosetEncodingSet }
        STRUCTURED WITH {
            ENCODING-SPACE
            EXHIBITS HANDLE «qn» AT { 0 | 1 | 2 | 3 }
            AS bits:'1111'B }}
    WITH FastInfosetEncodingSet }
-- Кодируют поле длины, которое было добавлено перед SEQUENCE
-- OF, и кодируют SEQUENCE OF NonEmptyOctetString (см. C.21)
eNonEmptySequenceOfWithLengthNonEmptyOctetString1{<#C>}
#PrecededByNonEmptySequenceOfLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptySequenceOfLength1,
        original {
            ENCODE STRUCTURE {
                eNonEmptyOctetStringPrepaddingAdder1
                STRUCTURED WITH eRepetitionItems1{<length>}
            } WITH PER-BASIC-UNALIGNED }}
    WITH FastInfosetEncodingSet }
-- Кодируют поле длины, которое было добавлено перед SEQUENCE
-- OF, и кодируют SEQUENCE OF EncodedCharacterString (см. C.21)
eNonEmptySequenceOfWithLengthEncodedCharacterString1{<#C>}
#PrecededByNonEmptySequenceOfLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptySequenceOfLength1,
        original {
            ENCODE STRUCTURE {
                eEncodedCharacterStringDriver1
                STRUCTURED WITH eRepetitionItems1{<length>}
            } WITH PER-BASIC-UNALIGNED }}
    WITH FastInfosetEncodingSet }
-- Кодируют поле длины, которое было добавлено перед SEQUENCE
-- OF, и кодируют SEQUENCE OF NameSurrogate (см. C.21)
eNonEmptySequenceOfWithLengthNameSurrogate1{<#C>}
#PrecededByNonEmptySequenceOfLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptySequenceOfLength1,
        original {
```

```
            ENCODE STRUCTURE {
                eNameSurrogateDriver1
                STRUCTURED WITH eRepetitionItems1{<length>}
            } WITH PER-BASIC-UNALIGNED }}
    WITH FastInfosetEncodingSet }
-- Кодируют поле длины, которое было добавлено перед SEQUENCE
-- OF, и кодируют SEQUENCE OF additional-datum (см. C.21)
eNonEmptySequenceOfWithLengthAdditionalDatum1{<#C>}
#PrecededByNonEmptySequenceOfLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptySequenceOfLength1,
        original {
            ENCODE STRUCTURE {
                additional-datum {
                    ENCODE STRUCTURE {
                        id eNonEmptyOctetStringPrepaddingAdder1,
                        data eNonEmptyOctetStringPrepaddingAdder1 }
                    WITH FastInfosetEncodingSet}
                    STRUCTURED WITH eRepetitionItems1{<length>}
            } WITH PER-BASIC-UNALIGNED }}
    WITH FastInfosetEncodingSet }
-- Кодируют поле длины, которое было добавлено перед
-- NonEmptyOctetString, и кодируют NonEmptyOctetString.
-- Используют, когда кодирование начинается со второго бита
-- октета (см. C.22)
eNonEmptyOctetStringWithLength2{<#C>}
#PrecededByNonEmptyOctetStringLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptyOctetStringLength2,
        original eOctetStringOctets1{<length>} }
    WITH FastInfosetEncodingSet }
-- Кодируют поле длины, которое было добавлено перед
-- NonEmptyOctetString, и кодируют NonEmptyOctetString.
-- Используют, когда кодирование начинается с пятого бита октета (см. C.23)
eNonEmptyOctetStringWithLength5{<#C>}
#PrecededByNonEmptyOctetStringLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptyOctetStringLength5,
        original eOctetStringOctets1{<length>} }
    WITH FastInfosetEncodingSet }
-- Кодируют поле длины, которое было добавлено перед
-- NonEmptyOctetString, и кодируют NonEmptyOctetString.
-- Используют, когда кодирование начинается с седьмого бита
-- октета (см. C.24)
eNonEmptyOctetStringWithLength7{<#C>}
#PrecededByNonEmptyOctetStringLength{<#C>} ::= {
    ENCODE STRUCTURE {
        length eNonEmptyOctetStringLength7,
        original eOctetStringOctets1{<length>} }
    WITH FastInfosetEncoingSet }
-- Кодируют дискриминант, который был добавлен перед элементом
-- компонента children типа Document, и кодируют этот элемент
-- (см. C.2.11.2 – C.2.11.5)
eDocumentChildWithDiscriminantIor5{<#C>}
#PrecededByFourAlternativeDiscriminant{<#C>} ::= {
```

```
    ENCODE STRUCTURE {
        prepadding {
            ALIGNED TO NEXT octet
            ENCODING-SPACE SIZE 0 },
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '11100001'B,
                2 TO '11100010'B,
                3 TO '110001'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                element eElement2,
                processing-instruction eProcessingInstruction1,
                comment eComment1,
                document-type-declaration eDocumentTypeDeclaration7
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед элементом
-- компонента children типа Element, и кодируют этот элемент
-- (см. C.3.7.2 – C.3.7.6)
eElementChildWithDiscriminant1or5{<#C>}
#PrecededByFiveAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        prepadding {
            ALIGNED TO NEXT octet
            ENCODING-SPACE SIZE 0 },
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '11100001'B,
                2 TO '110010'B,
                3 TO '10'B,
                4 TO '11100010'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                element eElement2,
                processing-instruction eProcessingInstruction1,
                unexpanded-entity-reference
                                        eUnexpandedEntityReference7,
                character-chunk eCharacterChunk3,
                comment eComment1
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед длиной
-- SEQUENCE OF (идентифицируя один из двух способов кодирования
```

```
-- длины), и кодируют эту длину (см. С.21)
eNonEmptySequenceOfLengthWithDiscriminant1{<#C>}
#PrecededByTwoAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '1000'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед длиной
-- NonEmptyOctetString (идентифицируя один из трех
-- способов кодирования длины), и кодируют длину. Используют,
-- когда кодирование начинается со второго бита октета (см. С.22)
eNonEmptyOctetStringLengthWithDiscriminant2{<#C>}
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '1000000'B,
                2 TO '1100000'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед длиной
-- NonEmptyOctetString (идентифицируя один из трех способов
-- кодирования длины), и кодируют длину. Используют, когда
-- кодирование начинается с пятого бита октета (см. С.23)
eNonEmptyOctetStringLengthWithDiscriminant5{<#C>}
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '1000'B,
                2 TO '1100'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
```

```
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед длиной
-- NonEmptyOctetString (идентифицируя один из трех способов
-- кодирования длины), и кодируют длину. Используют, когда
-- кодирование начинается с седьмого бита октета (см. С.24)
eNonEmptyOctetStringLengthWithDiscriminant7{<#C>}
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '10'B,
                2 TO '11'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед
-- положительным целым значением (идентифицируя один из трех
-- способов кодирования целого), и кодируют целое значение.
-- Используют, когда кодирование начинается со второго бита
-- октета (см. С.25)
ePositiveIntegerWithDiscriminant2{<#C>}
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '10'B,
                2 TO '110'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }
        STRUCTURED WITH {
            ENCODING-SPACE SIZE self-delimiting-values
            EXHIBITS HANDLE «qn» AT { 0 | 1 | 2 | 3 }
            AS range:{low 0, high 12}}} -- Меньше '1110'B
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед
-- положительным целым значением (идентифицируя один из четырех
-- способов кодирования целого), и кодируют целое значение.
-- Используют, когда кодирование начинается с третьего бита
-- октета (см. С.27)
```

```
ePositiveIntegerWithDiscriminant3{<#C>}
#PrecededByFourAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '100'B,
                2 TO '101'B,
                3 TO '110000000'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }
        STRUCTURED WITH {
            ENCODING-SPACE SIZE self-delimiting-values
            EXHIBITS HANDLE «qn» AT { 0 | 1 | 2 | 3 }
            AS range:{low 0, high 14}}} -- Меньше '1111'B
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед
-- положительным целым значением (идентифицируя один из четырех
-- способов кодирования целого), и кодируют целое значение.
-- Используют, когда кодирование начинается с четвертого бита
-- октета (см. С.28)
ePositiveIntegerWithDiscriminant4{<#C>}
#PrecededByFourAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '0'B,
                1 TO '100'B,
                2 TO '101'B,
                3 TO '110000000'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют дискриминант, который был добавлен перед
-- неотрицательным целым значением (идентифицируя один из трех
-- способов кодирования этого целого значения), и кодируют целое
-- значение (см. С.26)
eNonNegativeIntegerWithDiscriminant2{<#C>}
#PrecededByFourAlternativeDiscriminant{<#C>} ::= {
    ENCODE STRUCTURE {
        discriminant {
            USE #BIT-STRING
             MAPPING TO BITS {
                0 TO '1111111'B,
```

```
                1 TO '0'B,
                2 TO '10'B,
                3 TO '110'B }
            WITH FastInfosetEncodingSet },
        original {
            ENCODE STRUCTURE {
                STRUCTURED WITH {
                    ALTERNATIVE DETERMINED BY field-to-be-set
                    USING discriminant }}
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют тип Document (см. C.2)
eDocument2 #Document ::= {
    ENCODE STRUCTURE {
        additional-data {
           ENCODE STRUCTURE {
              STRUCTURED WITH eRepetitionWithLengthAdditionalDatum1 }
           WITH FastInfosetEncodingSet }
                OPTIONAL-ENCODING USE-SET,
        initial-vocabulary {
            ENCODE STRUCTURE {
                STRUCTURED WITH eInitialVocabularyPrepaddingAdder1 }
            WITH FastInfosetEncodingSet }
                OPTIONAL-ENCODING USE-SET,
        notations {
            ENCODE STRUCTURE {
                eNotationDriver1
                STRUCTURED WITH eRepetitionWithTerminator8bit1 }
            WITH FastInfosetEncodingSet }
                OPTIONAL-ENCODING USE-SET,
        unparsed-entities {
            ENCODE STRUCTURE {
                eUnparsedEntityDriver1
                STRUCTURED WITH eRepetitionWithTerminator8bit1 }
            WITH FastInfosetEncodingSet }
                OPTIONAL-ENCODING USE-SET,
        character-encoding-scheme
                eNonEmptyOctetStringPrepaddingAdder1
                OPTIONAL-ENCODING USE-SET,
        standalone eStandalonePrepaddingAdder1
                OPTIONAL-ENCODING USE-SET,
        version eNonIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
        children {
            ENCODE STRUCTURE {
                {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                        eDocumentChildDiscriminantAdder1or5 }
                    WITH FastInfosetEncodingSet }
                STRUCTURED WITH eRepetitionWithTerminator4bit1 }
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют тип Element (см. C.3)
eElement2 #Element ::= {
    ENCODE STRUCTURE {
```

```
            namespace-attributes  {
                ENCODE STRUCTURE {
                    eNamespaceAttributeDriver1
                    STRUCTURED WITH eRepetitionWithTerminator10bit1 }
                WITH FastInfosetEncodingSet }
                    OPTIONAL-ENCODING eNamespaceAttributesOptionality3,
            qualified-name eQualifiedNameOrIndex3,
            attributes {
                ENCODE STRUCTURE {
                    eAttributeDriver1
                    STRUCTURED WITH eRepetitionWithTerminator4bit1 }
                WITH FastInfosetEncodingSet }
                    OPTIONAL-ENCODING USE-SET,
            children {
                ENCODE STRUCTURE {
                    {
                        ENCODE STRUCTURE {
                            STRUCTURED WITH
                            eElementChildDiscriminantAdder1or5 }
                        WITH FastInfosetEncodingSet }
                    STRUCTURED WITH eRepetitionWithTerminator4bit1 }
                WITH FastInfosetEncodingSet }}
        WITH FastInfosetEncodingSet }
    -- Кодируют тип Attribute (см. С.4)
    eAttribute2 #Attribute ::= {
        ENCODE STRUCTURE {
            qualified-name eQualifiedNameOrIndex2,
            normalized-value eNonIdentifyingStringOrIndex1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют тип ProcessingInstruction (см. С.5)
    eProcessingInstruction1 #ProcessingInstruction ::= {
        ENCODE STRUCTURE {
            target eIdentifyingStringOrIndex1,
            content eNonIdentifyingStringOrIndex1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют тип UnexpandedEntityReference (см. С.6)
    eUnexpandedEntityReference7 #UnexpandedEntityReference ::= {
        ENCODE STRUCTURE {
            name eIdentifyingStringOrIndex1,
            system-identifier eIdentifyingStringOrIndex1
                    OPTIONAL-ENCODING USE-SET,
            public-identifier eIdentifyingStringOrIndex1
                    OPTIONAL-ENCODING USE-SET }
        WITH FastInfosetEncodingSet }
    -- Кодируют тип CharacterChunk (см. С.7)
    eCharacterChunk3 #CharacterChunk ::= {
        ENCODE STRUCTURE {
            character-codes eNonIdentifyingStringOrIndex3 }
        WITH FastInfosetEncodingSet }
    -- Кодируют тип Comment (см. С.8)
    eComment1 #Comment ::= {
        ENCODE STRUCTURE {
            content eNonIdentifyingStringOrIndex1 }
        WITH FastInfosetEncodingSet }
    -- Кодируют тип DocumentTypeDeclaration (см. С.9)
    eDocumentTypeDeclaration7 #DocumentTypeDeclaration ::= {
```

```
    ENCODE STRUCTURE {
        system-identifier eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
        public-identifier eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
        children {
            ENCODE STRUCTURE {
                {
                    ENCODE STRUCTURE {
                        STRUCTURED WITH
                        eDocTypeDeclChildPrepaddingAdder1 }
                    WITH FastInfosetEncodingSet }
                STRUCTURED WITH eRepetitionWithTerminator4bit1 }
            WITH FastInfosetEncodingSet }}
    WITH FastInfosetEncodingSet }
-- Кодируют тип UnparsedEntity (см. C.10)
eUnparsedEntity8 #UnparsedEntity ::= {
    ENCODE STRUCTURE {
        name eIdentifyingStringOrIndex1,
        system-identifier eIdentifyingStringOrIndex1,
        public-identifier eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
        notation-name eIdentifyingStringOrIndex1 }
    WITH FastInfosetEncodingSet }
-- Кодируют тип Notation (см. C.11)
eNotation7 #Notation ::= {
    ENCODE STRUCTURE {
        name eIdentifyingStringOrIndex1,
        system-identifier eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
        public-identifier eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET }
    WITH FastInfosetEncodingSet }
-- Кодируют тип NamespaceAttribute (см. C.12)
eNamespaceAttribute7 #NamespaceAttribute ::= {
    ENCODE STRUCTURE {
        prefix eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET,
        namespace-name eIdentifyingStringOrIndex1
                OPTIONAL-ENCODING USE-SET }
    WITH FastInfosetEncodingSet }
-- Кодируют тип IdentifyingStringOrIndex (см. C.13)
eIdentifyingStringOrIndex1 #IdentifyingStringOrIndex ::= {
    ENCODE STRUCTURE {
        literal-character-string eNonEmptyOctetString2,
        string-index ePositiveInteger2 }
    WITH FastInfosetEncodingSet }
-- Кодируют тип NonIdentifyingStringOrIndex. Используют, когда
-- кодирование начинается с первого бита октета (см. C.14)
eNonIdentifyingStringOrIndex1 #NonIdentifyingStringOrIndex ::= {
    ENCODE STRUCTURE {
        literal-character-string {
            ENCODE STRUCTURE {
                add-to-table USE-SET,
                character-string eEncodedCharacterString3 }
            WITH FastInfosetEncodingSet },
```

```
        string-index eNonNegativeInteger2 }
    WITH FastInfosetEncodingSet }
-- Кодируют тип NonIdentifyingStringOrIndex. Используют, когда
-- кодирование начинается с третьего бита октета (см. С.15)
eNonIdentifyingStringOrIndex3 #NonIdentifyingStringOrIndex ::= {
    ENCODE STRUCTURE {
        literal-character-string {
            ENCODE STRUCTURE {
                add-to-table USE-SET,
                character-string eEncodedCharacterString5 }
            WITH FastInfosetEncodingSet },
        string-index ePositiveInteger4 }
    WITH FastInfosetEncodingSet }
-- Кодируют тип NameSurrogate (см. С.16)
eNameSurrogate7 #NameSurrogate ::= {
    ENCODE STRUCTURE {
        prefix-string-index ePositiveInteger2
                OPTIONAL-ENCODING USE-SET,
        namespace-name-string-index ePositiveInteger2
                OPTIONAL-ENCODING USE-SET,
        local-name-string-index ePositiveInteger2 }
    WITH FastInfosetEncodingSet }
-- Кодируют тип QualifiedNameOrIndex. Используют, когда
-- кодирование начинается со второго бита октета (см. С.17)
eQualifiedNameOrIndex2 #QualifiedNameOrIndex ::= {
    ENCODE STRUCTURE {
        literal-qualified-name {
            ENCODE STRUCTURE {
                STRUCTURED WITH
                eLiteralQualifiedNamePrepaddingAdder2 }
            WITH FastInfosetEncodingSet },
        name-surrogate-index ePositiveInteger2
        STRUCTURED WITH eQualifiedNameAlternatives3 }
    WITH FastInfosetEncodingSet }
-- Кодируют тип QualifiedNameOrIndex. Используют, когда
-- кодирование начинается с третьего бита октета (см. С.18)
eQualifiedNameOrIndex3 #QualifiedNameOrIndex ::= {
    ENCODE STRUCTURE {
        literal-qualified-name {
            ENCODE STRUCTURE {
                STRUCTURED WITH
                eLiteralQualifiedNamePrepaddingAdder3 }
            WITH FastInfosetEncodingSet },
        name-surrogate-index ePositiveInteger3
        STRUCTURED WITH eQualifiedNameAlternatives3 }
    WITH FastInfosetEncodingSet }
-- Кодируют тип EncodedCharacterString. Используют, когда
-- кодирование начинается с третьего бита октета (см. С.19)
eEncodedCharacterString3 #EncodedCharacterString ::= {
    ENCODE STRUCTURE {
        encoding-format USE-SET,
        octets eNonEmptyOctetString5 }
    WITH FastInfosetEncodingSet }
-- Кодируют тип EncodedCharacterString. Используют, когда
-- кодирование начинается с пятого бита октета (см. С.20)
eEncodedCharacterString5 #EncodedCharacterString ::= {
```

```
    ENCODE STRUCTURE {
        encoding-format USE-SET,
        octets eNonEmptyOctetString7 }
    WITH FastInfosetEncodingSet }
-- Кодируют повторение (SEQUENCE OF NonEmptyOctetString),
-- вставляя перед ним поле длины (см. C.2.5.3 – C.2.5.5)
eRepetitionWithLengthNonEmptyOctetString1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
        ENCODED BY eNonEmptySequenceOfWithLengthNonEmptyOctetString1
}}
-- Кодируют повторение (SEQUENCE OF EncodedCharacterString),
-- вставляя перед ним поле длины (см. C.2.5.3 – C.2.5.5)
eRepetitionWithLengthEncodedCharacterString1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
        ENCODED BY
        eNonEmptySequenceOfWithLengthEncodedCharacterString1 }}
-- Кодируют повторение (SEQUENCE OF NameSurrogate), вставляя
-- перед ним поле длины (см. C.2.5.3 – C.2.5.5)
eRepetitionWithLengthNameSurrogate1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
        ENCODED BY eNonEmptySequenceOfWithLengthNameSurrogate1 }}
-- Кодируют повторение (SEQUENCE OF additional-datum), вставляя
-- перед ним поле длины (см. C.2.5.3 – C.2.5.5)
eRepetitionWithLengthAdditionalDatum1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
        ENCODED BY eNonEmptySequenceOfWithLengthAdditionalDatum1 }}
-- Кодируют тип NonEmptyOctetString. Используют, когда
-- кодирование начинается со второго бита октета (см. C.22)
eNonEmptyOctetString2 #NonEmptyOctetString ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
        ENCODED BY eNonEmptyOctetStringWithLength2 }}
-- Кодируют тип NonEmptyOctetString. Используют, когда
-- кодирование начинается с пятого бита октета (см. C.23)
eNonEmptyOctetString5 #NonEmptyOctetString ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
        ENCODED BY eNonEmptyOctetStringWithLength5 }}
-- Кодируют тип NonEmptyOctetString. Используют, когда
-- кодирование начинается с седьмого бита октета (см. C.24)
eNonEmptyOctetString7 #NonEmptyOctetString ::= {
    REPETITION-ENCODING {
        REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
        ENCODED BY eNonEmptyOctetStringWithLength7 }}
-- Кодируют поле длины, вставленное перед кодированием
-- SEQUENCE OF (см. C.21)
eNonEmptySequenceOfLength1 #NonEmptySequenceOfLength ::= {
    USE #NonEmptySequenceOfLengthAlternatives1
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH
```

```
            eNonEmptySequenceOfLengthDiscriminantAdder1 }
        WITH FastInfosetEncodingSet }}
-- Кодируют поле длины, вставленное перед кодированием
-- NonEmptyOctetString. Используют, когда кодирование начинается
-- со второго бита октета (см. С.22)
eNonEmptyOctetStringLength2 #NonEmptyOctetStringLength ::= {
    USE #NonEmptyOctetStringLengthAlternatives2
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH
            eNonEmptyOctetStringLengthDiscriminantAdder2 }
        WITH FastInfosetEncodingSet }}
-- Кодируют поле длины, вставленное перед кодированием
-- NonEmptyOctetString. Используют, когда кодирование начинается
-- с пятого бита октета (см. С.23)
eNonEmptyOctetStringLength5 #NonEmptyOctetStringLength ::= {
    USE #NonEmptyOctetStringLengthAlternatives5
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH
            eNonEmptyOctetStringLengthDiscriminantAdder5 }
        WITH FastInfosetEncodingSet }}
-- Кодируют поле длины, вставленное перед кодированием
-- NonEmptyOctetString. Используют, когда кодирование начинается
-- с седьмого бита октета (см. С.24)
eNonEmptyOctetStringLength7 #NonEmptyOctetStringLength ::= {
    USE #NonEmptyOctetStringLengthAlternatives7
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH
            eNonEmptyOctetStringLengthDiscriminantAdder7 }
        WITH FastInfosetEncodingSet }}
-- Кодируют положительное целое значение. Используют, когда
-- кодирование начинается со второго бита октета (см. С.25)
ePositiveInteger2 #PositiveOrNonNegativeInteger ::= {
    USE #PositiveIntegerAlternatives2
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH ePositiveIntegerDiscriminantAdder2 }
        WITH FastInfosetEncodingSet }}
-- Кодируют положительное целое значение. Используют, когда
-- кодирование начинается с третьего бита октета (см. С.27)
ePositiveInteger3 #PositiveOrNonNegativeInteger ::= {
    USE #PositiveIntegerAlternatives3
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH ePositiveIntegerDiscriminantAdder3 }
        WITH FastInfosetEncodingSet }}
-- Кодируют положительное целое значение. Используют, когда
-- кодирование начинается с четвертого бита октета (см. С.28)
ePositiveInteger4 #PositiveOrNonNegativeInteger ::= {
```

```
    USE #PositiveIntegerAlternatives4
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH ePositiveIntegerDiscriminantAdder4 }
        WITH FastInfosetEncodingSet }}
-- Кодируют неотрицательное целое значение (см. С.26)
eNonNegativeInteger2 #PositiveOrNonNegativeInteger ::= {
    USE #NonNegativeIntegerAlternatives2
    MAPPING ORDERED VALUES
    WITH {
        ENCODE STRUCTURE {
            STRUCTURED WITH eNonNegativeIntegerDiscriminantAdder2 }
        WITH FastInfosetEncodingSet }}
-- Специфицируют, как определить наличие компонента
-- namespace-attributes типа Element (см. С.3.4.2)
eNamespaceAttributesOptionality3 #OPTIONAL ::= {
    PRESENCE DETERMINED BY handle
    HANDLE «nsa» }
-- Специфицируют, как определить альтернативу типа
-- QualifiedNameOrIndex (см. С.17.3 и С.18.3)
eQualifiedNameAlternatives3 #ALTERNATIVES ::= {
    ALTERNATIVE DETERMINED BY handle
    HANDLE «qn»
    EXHIBITS HANDLE «nsa» AT { 0 | 1 | 2 | 3 | 4 | 5}
    AS range:{low 0, high 50}} -- Меньше '110011'B
-- Специфицируют, как определить окончание повторения, используя
-- 4-битный ограничитель '1111' (см. С.2.12, С.3.6.2, С.3.8 и С.9.7)
eRepetitionWithTerminator4bit1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant
        DETERMINED BY pattern PATTERN bits:'1111'B }}
-- Специфицируют, как определить окончание повторения, используя
-- 8-битный ограничитель '11110000' (см. С.2.6.2 и С.2.7.2)
eRepetitionWithTerminator8bit1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant
        DETERMINED BY pattern PATTERN bits:'11110000'B }}
-- Специфицируют, как определить окончание повторения, используя
-- 10-битный ограничитель '1111000000' (см. С.3.4.3)
eRepetitionWithTerminator10bit1 #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant
        DETERMINED BY pattern PATTERN bits:'1111000000'B
        EXHIBITS HANDLE «nsa» AT { 0 | 1 | 2 | 3 | 4 | 5}
        AS bits:'110011'B }}
-- Кодируют элементы SEQUENCE OF, следующие за добавленным
-- полем длины (см. С.2.5.3 – С.2.5.5)
eRepetitionItems1{<REFERENCE:len>} #REPETITION ::= {
    REPETITION-ENCODING {
        REPETITION-SPACE SIZE variable-with-determinant
        MULTIPLE OF bit
        DETERMINED BY field-to-be-set USING len }}
-- Кодируют октеты NonEmptyOctetString, следующие за добавленным
-- полем длины (см. С.22, С.23 и С.24)
eOctetStringOctets1{<REFERENCE:len>} #OCTETS ::= {
```

```
        REPETITION-ENCODING {
            REPETITION-SPACE SIZE variable-with-determinant
            MULTIPLE OF bit
            DETERMINED BY field-to-be-set USING len }}
    empty-padding #PAD ::= {
        ENCODING-SPACE SIZE 0
    }
    FastInfosetEncodingSet #ENCODINGS::= { eDocument2 | empty-padding }
        COMPLETED BY PER-BASIC-UNALIGNED
    END
    FastInfosetELM
        {joint-iso-itu-t(2) asn1(1) generic-applications(10)
        fast-infoset(0) modules(0) fast-infoset-elm(2)}
        LINK-DEFINITIONS ::= BEGIN
        IMPORTS FastInfosetEncodingSet, Document FROM FastInfosetEDM;
        ENCODE #Document WITH FastInfosetEncodingSet
    END
```

Приложение В
(обязательное)

MIME-медиатип для документов быстрого инфо-набора

В настоящем приложении определен медиатип «application/fastinfoset», который описывает документы быстрого инфо-набора.

Данный MIME-медиатип определен ниже с помощью IETF MIME-шаблона регистрации. Зарегистрирован он в соответствии с процедурами IETF.

Название MIME-медиатипа:
application
Название MIME-подтипа:
fastinfoset
Обязательные параметры:
Нет.
Необязательные параметры:
Нет.
Аспекты кодирования:

Инфо-наборы XML, закодированные как документы быстрого инфо-набора, приводят к созданию двоичных данных. Данный MIME-тип может потребовать последующее кодирование при передаче, не поддерживающей двоичные данные.

Аспекты безопасности:

Так как инфо-наборы XML, закодированные как документы быстрого инфо-набора, могут переносить определенные приложением данные, семантика которых не зависит от семантики MIME-оболочки (или контекста, в котором используется MIME-оболочка), не следует ожидать, что семантику документа быстрого инфо-набора можно понять на основе семантики только MIME-оболочки. Следовательно, при использовании медиатипа «application/fastinfoset» настоятельно рекомендуется, чтобы вопросы безопасности контекста, в котором используют документ быстрого инфо-набора, были полностью понятны.

Аспекты совместимости:
Известных проблем совместимости не существует.
Опубликованная спецификация:
Рек. МСЭ-Т X.891 | ISO/IEC 24824-1
Приложения, использующие данный медиатип:
Нет известных приложений, которые используют данный медиатип.
Дополнительная информация:
Магическое(ие) число(а):

Документ быстрого инфо-набора может начинаться с опциональной декларации XML, которая должна быть одной из следующих строк, закодированных в UTF-8:

```
<?xml encoding='finf'?>
<?xml encoding='finf' standalone='yes'?>
<?xml encoding='finf' standalone='no'?>
<?xml version='1.0' encoding='finf'?>
<?xml version='1.0' encoding='finf' standalone='yes'?>
<?xml version='1.0' encoding='finf' standalone='no'?>
<?xml version='1.1' encoding='finf'?>
<?xml version='1.1' encoding='finf' standalone='yes'?>
<?xml version='1.1' encoding='finf' standalone='no'?>
```

Первые пять октетов декларации XML, закодированной в UTF-8, являются шестнадцатеричными цифрами 3C 3F 78 6D 6C. Четыре октета, идентифицирующие документ быстрого инфо-набора, соответствующие закодированной в UTF-8 подстроке «finf», являются шестнадцатеричными цифрами 66 69 6E 66.

Документ быстрого инфо-набора должен начинаться с шестнадцатеричной последовательности октетов E0 00 00 01, если опциональная декларация XML отсутствует.

Расширение(я) файла(ов):

*.finf

Контактное лицо и адрес электронной почты для получения дополнительной информации:

ITU-T ASN.1 докладчик (gost.e-ecolog.ru

ISO/IEC JTC1/SC6 ASN.1 докладчик (gost.e-ecolog.ru

Предполагаемое использование:

ОБЩЕЕ

Автор/Ответственный за изменения:

Совместные процедуры голосования ITU-T и ISO/IEC в соответствии с Рек. МСЭ-Т А.23 *Сотрудничество с Международной организацией по стандартизации (ISO) и Международной электротехнической комиссией (IEC) по вопросам информационных технологий*, приложение А и Директивы ISO/IEC JTC1, приложение К.

Приложение С
(справочное)

Описание кодирования документа быстрого инфо-набора

С.1 Документ быстрого инфо-набора

С.1.1 В настоящем приложении неформально (но точно и полно) описаны кодирования, определенные в разделе 12 и приложении А. Для удобства реализации все определения типов ASN.1, приведенные в нормативном тексте, скопированы в данном приложении.

С.1.2 Кодирования описаны в терминах действий, осуществляемых кодировщиком, в результате которых биты добавляются к потоку битов. Действия, которые должны выполняться декодером, не описаны в явном виде в настоящем приложении, но могут быть выведены на основе описанных в настоящем приложении действий кодировщика.

С.1.3 Документ быстрого инфо-набора может начинаться либо с декларации XML (см. 12.3), за которой следуют:

a) шестнадцать битов '1110000000000000' (идентификация),

b) шестнадцать битов '0000000000000001' (номер версии),

c) бит '0' (забивка)

либо с этих же тридцати трех битов без предшествующей им декларации XML. Непосредственно после тридцати трех битов следует кодирование абстрактного значения типа **Document**, как описано в С.2. Это кодирование заканчивается на восьмом или четвертом бите октета в зависимости от содержимого документа быстрого инфо-набора. В последнем случае к потоку битов добавляют четыре бита '0000' (забивка).

С.2 Кодирование типа Document

С.2.1 Тип **Document** определен в 7.2 следующим образом:

```
Document ::= SEQUENCE {
     additional-data SEQUENCE (SIZE(1..one-meg)) OF
          additional-datum SEQUENCE {
               id URI,
               data NonEmptyOctetString } OPTIONAL,
     initial-vocabulary SEQUENCE {
          external-vocabulary URI OPTIONAL,
          restricted-alphabets SEQUENCE (SIZE(1..256)) OF
              NonEmptyOctetString OPTIONAL,
          encoding-algorithms SEQUENCE (SIZE(1..256)) OF
               NonEmptyOctetString OPTIONAL,
          prefixes SEQUENCE (SIZE(1..one-meg)) OF
               NonEmptyOctetString OPTIONAL,
          namespace-names SEQUENCE (SIZE(1..one-meg)) OF
               NonEmptyOctetString OPTIONAL,
          local-names SEQUENCE (SIZE(1..one-meg)) OF
               NonEmptyOctetString OPTIONAL,
          other-ncnames SEQUENCE (SIZE(1..one-meg)) OF
               NonEmptyOctetString OPTIONAL,
          other-uris SEQUENCE (SIZE(1..one-meg)) OF
               NonEmptyOctetString OPTIONAL,
          attribute-values SEQUENCE (SIZE(1..one-meg)) OF
               EncodedCharacterString OPTIONAL,
          content-character-chunks SEQUENCE (SIZE(1..one-meg)) OF
               EncodedCharacterString OPTIONAL,
          other-strings SEQUENCE (SIZE(1..one-meg)) OF
               EncodedCharacterString OPTIONAL,
```

```
        element-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
             NameSurrogate OPTIONAL,
        attribute-name-surrogates SEQUENCE (SIZE(1..one-meg)) OF
             NameSurrogate OPTIONAL }
             CONSTRAINED BY {
                  -- Если присутствует компонент initial-
                  -- vocabulary, то должен присутствовать как
                  -- минимум один из его компонентов -- })
                  OPTIONAL,
   notations SEQUENCE (SIZE(1..MAX)) OF
                  Notation OPTIONAL,
   unparsed-entities SEQUENCE (SIZE(1..MAX)) OF
                  UnparsedEntity OPTIONAL,
   character-encoding-scheme NonEmptyOctetString OPTIONAL,
   standalone BOOLEAN OPTIONAL,
   version NonIdentifyingStringOrIndex OPTIONAL
        -- Категория OTHER STRING --,
   children SEQUENCE (SIZE(0..MAX)) OF
        CHOICE {
             element Element,
             processing-instruction ProcessingInstruction,
             comment Comment,
             document-type-declaration DocumentTypeDeclaration }}
```

С.2.2 Значение типа **Document** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинается со второго бита октета и заканчивается на четвертом или восьмом бите другого октета (который является последним битом указателя конца '1111', описанного в С.2.12).

С.2.3 Для каждого из семи опциональных компонентов **additional-data**, **initial-vocabulary**, **notations**, **unparsed-entities**, **character-encoding-scheme**, **standalone** и **version** (в указанном порядке) при наличии компонента к потоку битов добавляют бит '1' (наличие), в противном случае добавляют бит '0' (отсутствие).

С.2.4 Если опциональный компонент **additional-data** присутствует, то количество компонентов **additional-datum** кодируют, как описано в С.21, и каждый из компонентов **additional-datum** кодируют, как описано в двух следующих подпунктах.

С.2.4.1 Бит '0' (забивка) добавляют к потоку битов, и компонент **id** кодируют, как описано в С.22.

С.2.4.2 Бит '0' (забивка) добавляют к потоку битов, и компонент **data** кодируют, как описано в С.22.

С.2.5 Если опциональный компонент **initial-vocabulary** присутствует, то к потоку битов добавляют три бита '000' (забивка) и этот компонент кодируют, как описано в пяти следующих пунктах.

С.2.5.1 Для каждого из тринадцати опциональных компонентов компонента **initial-vocabulary** (в текстуальном порядке) при его наличии к потоку битов добавляют бит '1' (наличие), в противном случае добавляют бит '0' (отсутствие).

С.2.5.2 При наличии опционального компонента **external-vocabulary** компонента **initial-vocabulary** к потоку битов добавляют бит '0' (забивка) и компонент кодируют, как описано в С.22.

С.2.5.3 Для каждого из компонентов **restricted-alphabets**, **encoding-algorithms**, **prefixes**, **namespace-names**, **local-names**, **other-ncnames** и **other-uris** (в указанном порядке) при его наличии кодируют число элементов **NonEmptyOctetString** в компоненте, как описано в С.21, и затем каждый элемент кодируют (по порядку) следующим образом: к потоку битов добавляют бит '0' (забивка) и **NonEmptyOctetString** кодируют, как описано в С.22.

С.2.5.4 Для каждого из компонентов **attribute-values**, **content-character-chunks** и **other-strings** (в указанном порядке) при его наличии кодируют число элементов **EncodedCharacterString** в компоненте, как описано в С.21, и затем каждый элемент кодируют (по порядку) следующим образом: к потоку битов добавляют два бита '00' (забивка) и **EncodedCharacterString** кодируют, как описано в С.19.

С.2.5.5 Для каждого из компонентов **element-name-surrogates** и **attribute-name-surrogates** (в указанном порядке) при его наличии кодируют число элементов **NameSurrogate** в

компоненте, как описано в С.21, и затем каждый элемент кодируют (по порядку) следующим образом: к потоку битов добавляют шесть битов '000000' (забивка) и **NameSurrogate** кодируют, как описано в С.16.

С.2.6 При наличии опционального компонента **notations** его кодируют, как описано в двух следующих пунктах.

С.2.6.1 Каждый элемент **notations** (по порядку) кодируют следующим образом: к потоку битов добавляют шесть битов '110000' (идентификация) и **Notation** кодируют, как описано в С.11.

С.2.6.2 К потоку битов добавляют четыре бита '1111' (указатель конца) и четыре бита '0000' (забивка).

Примечание — Данные биты не добавляют, если компонент **notations** отсутствует.

С.2.7 При наличии опционального компонента **unparsed-entities** его кодируют так, как описано в двух следующих подпунктах.

С.2.7.1 Каждый элемент **unparsed-entities** (по порядку) кодируют следующим образом: к потоку битов добавляют семь битов '1101000' (идентификация) и **UnparsedEntity** кодируют, как описано в С.10.

С.2.7.2 К потоку битов добавляют четыре бита '1111' (указатель конца) и четыре бита '0000' (забивка).

Примечание — Данные биты не добавляют, если компонент **unparsed-entities** отсутствует.

С.2.8 При наличии опционального компонента **character-encoding-scheme** к потоку битов добавляют бит '0' (забивка) и **NonEmptyOctetString** кодируют, как описано в С.22.

С.2.9 При наличии опционального компонента **standalone** его кодируют следующим образом: к потоку битов добавляют семь битов '0000000' (забивка). Если значением **standalone** является **TRUE**, то к потоку битов добавляют бит '1', в противном случае — бит '0'.

С.2.10 При наличии опционального компонента **version** его значение кодируют, как описано в С.14.

С.2.11 Если компонент **children** имеет один или несколько элементов, то каждый элемент кодируют (по порядку), как описано в следующих пяти подпунктах.

С.2.11.1 Кодирование каждого элемента обязательно начинают с первого бита октета. Однако последний добавленный бит может быть восьмым или четвертым битом октета. Если это был четвертый бит октета, то к потоку битов добавляют биты '0000' (забивка), чтобы кодирование элемента начиналось с первого бита следующего октета.

С.2.11.2 При наличии альтернативы **element** к потоку битов добавляют бит '0' (идентификация) и **element** кодируют, как описано в С.3.

С.2.11.3 При наличии альтернативы **processing-instruction** к потоку битов добавляют восемь битов '11100001' (идентификация) и **processing-instruction** кодируют, как описано в С.5.

С.2.11.4 При наличии альтернативы **comment** к потоку битов добавляют восемь битов '11100010' (идентификация) и **comment** кодируют, как описано в С.8.

С.2.11.5 При наличии альтернативы **document-type-declaration** к потоку битов добавляют шесть битов '110001' (идентификация) и **document-type-declaration** кодируют так, как описано в С.9.

С.2.12 Добавляют четыре бита '1111' (указатель конца).

Примечание — Данные биты не добавляют, если компонент **children** не имеет элементов.

С.3 Кодирование типа Element

С.3.1 Тип **Element** определен в 7.3 следующим образом:

```
Element ::= SEQUENCE {
     namespace-attributes  SEQUENCE (SIZE(1..MAX)) OF
          NamespaceAttribute OPTIONAL,
     qualified-name QualifiedNameOrIndex
               -- Категория ELEMENT NAME --,
     attributes SEQUENCE (SIZE(1..MAX)) OF
          Attribute OPTIONAL,
     children SEQUENCE (SIZE(0..MAX)) OF
          CHOICE {
               element Element,
```

```
processing-instruction ProcessingInstruction,
unexpanded-entity-reference UnexpandedEntityReference,
character-chunk CharacterChunk,
comment Comment }}
```

С.3.2 Значение типа **Element** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование этого типа всегда начинают со второго бита октета и заканчивают на четвертом или восьмом бите другого октета (который является последним битом указателя конца '1111', описанного в С.3.8).

С.3.3 При наличии опционального компонента **attributes** к потоку битов добавляют бит '1' (наличие), в противном случае добавляют бит '0' (отсутствие).

С.3.4 При наличии опционального компонента **namespace-attributes** его кодируют, как описано в трех следующих подпунктах.

С.3.4.1 К потоку битов добавляют четыре бита '1110' (наличие) и два бита '00' (забивка).

С.3.4.2 Каждый элемент **namespace-attributes** (по порядку) кодируют следующим образом: к потоку битов добавляют шесть битов '110011' (идентификация) и **NamespaceAttribute** кодируют, как описано в С.12.

С.3.4.3 Добавляют четыре бита '1111' (указатель конца) и шесть битов '000000' (забивка).

Примечание — Данные биты не добавляют, если компонент **namespace-attributes** отсутствует.

С.3.5 Значение компонента **qualified-name** кодируют, как описано в С.18.

С.3.6 При наличии опционального компонента **attributes** его кодируют, как описано в двух следующих подпунктах.

С.3.6.1 Каждый элемент **attributes** (по порядку) кодируют следующим образом: к потоку битов добавляют бит '0' (идентификация) и **Attribute** кодируют, как описано в С.4.

С.3.6.2 Добавляют четыре бита '1111' (указатель конца).

Примечание — Данные биты не добавляют, если компонент **attributes** отсутствует.

С.3.7 Если компонент **children** имеет один или несколько элементов, то каждый элемент кодируют (по порядку), как описано в шести следующих подпунктах.

С.3.7.1 Кодирование каждого элемента обязательно начинают с первого бита октета. Однако последний добавленный бит может быть восьмым или четвертым битом октета. Если это был четвертый бит октета, то к потоку битов добавляют биты '0000' (забивка), чтобы кодирование элемента начиналось с первого бита следующего октета.

С.3.7.2 При наличии альтернативы **element** к потоку битов добавляют бит '0' (идентификация) и **element** кодируют, как описано в С.3.

С.3.7.3 При наличии альтернативы **processing-instruction** к потоку битов добавляют восемь битов '11100001' (идентификация) и **processing-instruction** кодируют, как описано в С.5.

С.3.7.4 При наличии альтернативы **unexpanded-entity-reference** к потоку битов добавляют шесть битов '110010' (идентификация) и **unexpanded-entity-reference** кодируют, как описано в С.6.

С.3.7.5 При наличии альтернативы **character-chunk** к потоку битов добавляют два бита '10' (идентификация) и **character-chunk** кодируют, как описано в С.7.

С.3.7.6 При наличии альтернативы **comment** к потоку битов добавляют восемь битов '11100010' (идентификация) и **comment** кодируют, как описано в С.8.

С.3.8 Добавляют четыре бита '1111' (указатель конца).

Примечание — Данные биты не добавляют, если компонент **children** не имеет элементов.

С.4 Кодирование типа Attribute

С.4.1 Тип **Attribute** определен в 7.4 следующим образом:

```
Attribute ::= SEQUENCE {
     qualified-name QualifiedNameOrIndex
               -- Категория ATTRIBUTE NAME --,
     normalized-value NonIdentifyingStringOrIndex
               -- Категория ATTRIBUTE VALUE -- }
```

C.4.2 Значение типа **Attribute** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают со второго бита октета и заканчивают на восьмом бите другого октета.

C.4.3 Значение **qualified-name** кодируют так, как описано в C.17.
C.4.4 Значение **normalized-value** кодируют так, как описано в C.14.

C.5 Кодирование типа ProcessingInstruction

C.5.1 Тип **ProcessingInstruction** определен в 7.5 следующим образом:

```
ProcessingInstruction ::= SEQUENCE {
     target IdentifyingStringOrIndex
          -- Категория OTHER NCNAME --,
     content NonIdentifyingStringOrIndex
          -- Категория OTHER STRING -- }
```

C.5.2 Значение типа **ProcessingInstruction** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают с первого бита октета и заканчивают на восьмом бите другого октета.

C.5.3 Значение **target** кодируют, как описано в C.13.
C.5.4 Значение **content** кодируют, как описано в C.14.

C.6 Кодирование типа UnexpandedEntityReference

C.6.1 Тип **UnexpandedEntityReference** определен в 7.6. следующим образом:

```
UnexpandedEntityReference ::= SEQUENCE {
     name IdentifyingStringOrIndex
               -- Категория OTHER NCNAME --,
     system-identifier IdentifyingStringOrIndex OPTIONAL
               -- Категория OTHER URI --,
     public-identifier IdentifyingStringOrIndex OPTIONAL
               -- Категория OTHER URI -- }
```

C.6.2 Значение типа **UnexpandedEntityReference** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают с седьмого бита октета и заканчивают на восьмом бите другого октета.

C.6.3 При наличии опциональных компонентов **system-identifier** и **public-identifier** (в указанном порядке) к потоку битов добавляют бит '1' (наличие) для каждого из них, в противном случае добавляют бит '0' (отсутствие).
C.6.4 Значение **name** кодируют, как описано в C.13.
C.6.5 При наличии опционального компонента **system-identifier** его кодируют, как описано в C.13.
C.6.6 При наличии опционального компонента **public-identifier** его кодируют, как описано в C.13.

C.7 Кодирование типа CharacterChunk

C.7.1 Тип **CharacterChunk** определен в 7.7 следующим образом:

```
CharacterChunk ::= SEQUENCE {
     character-codes NonIdentifyingStringOrIndex
               -- Категория CONTENT CHARACTER CHUNK -- }
```

C.7.2 Значение типа **CharacterChunk** кодируют, выполняя следующее действие.

Примечание — Кодирование данного типа всегда начинают с третьего бита октета и заканчивают на восьмом бите другого или того же октета.

С.7.3 Значение **character-codes** кодируют, как описано в С.15.

С.8 Кодирование типа Comment

С.8.1 Тип **Comment** определен в 7.8 следующим образом:

```
Comment ::= SEQUENCE {
     content NonIdentifyingStringOrIndex
               -- Категория OTHER STRING --}
```

С.8.2 Значение типа **Comment** кодируют, выполняя следующее действие.

Примечание — Кодирование данного типа всегда начинают с первого бита октета и заканчивают на восьмом бите другого или того же октета.

С.8.3 Значение **content** кодируют, как описано в С.14.

С.9 Кодирование типа DocumentTypeDeclaration

С.9.1 Тип **DocumentTypeDeclaration** определен в 7.9 следующим образом:

```
DocumentTypeDeclaration ::= SEQUENCE {
     system-identifier IdentifyingStringOrIndex OPTIONAL
                    -- Категория OTHER URI --,
     public-identifier IdentifyingStringOrIndex OPTIONAL
                    -- Категория OTHER URI --,
     children SEQUENCE (SIZE(0..MAX)) OF
          ProcessingInstruction }
```

С.9.2 Значение типа **DocumentTypeDeclaration** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают с седьмого бита октета и заканчивают на четвертом бите другого октета (являющемся последним битом указателя конца '1111', описанного в С.9.7).

С.9.3 При наличии опциональных компонентов **system-identifier** и **public-identifier** (в указанном порядке) к потоку битов добавляют бит '1' (наличие) для каждого из них, в противном случае добавляют бит '0' (отсутствие).

С.9.4 При наличии опционального компонента **system-identifier** его кодируют, как описано в С.13.

С.9.5 При наличии опционального компонента **public-identifier** его кодируют, как описано в С.13.

С.9.6 Если компонент **children** имеет один или несколько элементов, то каждый элемент кодируют следующим образом: к потоку битов добавляют восемь битов '11100001' (идентификация), и **ProcessingInstruction** кодируют, как описано в С.5.

С.9.7 Добавляют четыре бита '1111' (указатель конца).

Примечание — Данные биты не добавляют, если компонент **children** не имеет элементов.

С.10 Кодирование типа UnparsedEntity

С.10.1 Тип **UnparsedEntity** определен в 7.10 следующим образом:

```
UnparsedEntity ::= SEQUENCE {
     name IdentifyingStringOrIndex
                    -- Категория OTHER NCNAME --,
     system-identifier IdentifyingStringOrIndex
                    -- Категория OTHER URI --,
     public-identifier IdentifyingStringOrIndex OPTIONAL
```

```
                    -- Категория OTHER URI --,
    notation-name IdentifyingStringOrIndex
                    -- Категория OTHER NCNAME -- }
```

С.10.2 Значение типа **UnparsedEntity** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают с восьмого бита октета и заканчивают на восьмом бите другого октета.

С.10.3 При наличии опционального компонента **public-identifier** к потоку битов добавляют бит '1' (наличие), в противном случае добавляют бит '0' (отсутствие).

С.10.4 Значение **name** кодируют, как описано в С.13.

С.10.5 Значение **system-identifier** кодируют, как описано в С.13.

С.10.6 При наличии опционального компонента **public-identifier** его кодируют, как описано в С.13.

С.10.7 Значение **name** кодируют, как описано в С.13.

С.11 Кодирование типа Notation

С.11.1 Тип **Notation** определен в 7.11 следующим образом:

```
Notation ::= SEQUENCE {
    name IdentifyingStringOrIndex
                    -- Категория OTHER NCNAME --,
    system-identifier IdentifyingStringOrIndex OPTIONAL
                    -- Категория OTHER URI --,
    public-identifier IdentifyingStringOrIndex OPTIONAL
                    -- Категория OTHER URI -- }
```

С.11.2 Значение типа **Notation** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают с седьмого бита октета и заканчивают на восьмом бите другого октета.

С.11.3 При наличии опциональных компонентов **system-identifier** и **public-identifier** (в указанном порядке) к потоку битов добавляют бит '1' (наличие) для каждого из них, в противном случае добавляют бит '0' (отсутствие).

С.11.4 Значение **name** кодируют, как описано в С.13.

С.11.5 При наличии опционального компонента **system-identifier** его кодируют, как описано в С.13.

С.11.6 При наличии опционального компонента **public-identifier** его кодируют, как описано в С.13.

С.12 Кодирование типа NamespaceAttribute

С.12.1 Тип **NamespaceAttribute** определен в 7.12 следующим образом:

```
NamespaceAttribute ::= SEQUENCE {
    prefix IdentifyingStringOrIndex OPTIONAL
                    -- Категория PREFIX --,
    namespace-name IdentifyingStringOrIndex OPTIONAL
                    -- Категория NAMESPACE NAME -- }
```

С.12.2 Значение типа **NamespaceAttribute** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают с восьмого бита октета и заканчивают на восьмом бите другого октета.

С.12.3 При наличии опционального компонента **prefix** к потоку битов добавляют бит '1' (наличие), в противном случае добавляют бит '0' (отсутствие).

C.12.4 При наличии опционального компонента **namespace-name** к потоку битов добавляют бит '1' (наличие), в противном случае добавляют бит '0' (отсутствие).

C.12.5 При наличии опционального компонента **prefix** его кодируют, как описано в C.13.

C.12.6 При наличии опционального компонента **namespace-name** его кодируют, как описано в C.13.

C.13 Кодирование типа IdentifyingStringOrIndex

C.13.1 Тип **IdentifyingStringOrIndex** определен в 7.13 следующим образом:

```
IdentifyingStringOrIndex ::= CHOICE {
     literal-character-string NonEmptyOctetString,
     string-index INTEGER (1..one-meg) }
```

C.13.2 Значение типа **IdentifyingStringOrIndex** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают с первого бита октета и заканчивают на восьмом бите другого или того же октета.

C.13.3 При наличии альтернативы **literal-character-string** к потоку битов добавляют бит '0' (дискриминант) и **literal-character-string** кодируют, как описано в C.22.

C.13.4 При наличии альтернативы **string-index** к потоку битов добавляют бит '1' (дискриминант) и **string-index** кодируют, как описано в C.25.

C.14 Кодирование типа NonIdentifyingStringOrIndex, начиная с первого бита октета

C.14.1 Тип **NonIdentifyingStringOrIndex** определен в 7.14 следующим образом:

```
NonIdentifyingStringOrIndex ::= CHOICE {
     literal-character-string SEQUENCE {
          add-to-table BOOLEAN,
          character-string EncodedCharacterString },
     string-index INTEGER (0..one-meg) }
```

C.14.2 В данном пункте (C.14) описано кодирование значения типа **NonIdentifyingStringOrIndex**, когда кодирование должно начаться с первого бита октета (см. также C.15). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого или того же октета.

C.14.3 При наличии альтернативы **literal-character-string** к потоку битов добавляют бит '0' (дискриминант) и **literal-character-string** кодируют, как описано в двух следующих подпунктах.

C.14.3.1 Если значение компонента **add-to-table** равно **TRUE**, то к потоку битов добавляют бит '1', в противном случает добавляют бит '0'.

C.14.3.2 Значение компонента **character-string** кодируют, как описано в C.19.

C.14.4 При наличии альтернативы **string-index** к потоку битов добавляют бит '1' (дискриминант) и **string-index** кодируют, как описано в C.26.

C.15 Кодирование типа NonIdentifyingStringOrIndex, начиная с третьего бита октета

C.15.1 Тип **NonIdentifyingStringOrIndex** определен в 7.14 следующим образом:

```
NonIdentifyingStringOrIndex ::= CHOICE {
     literal-character-string SEQUENCE {
          add-to-table BOOLEAN,
          character-string EncodedCharacterString },
     string-index INTEGER (0..one-meg) }
```

С.15.2 В данном пункте (С.15) описано кодирование значения типа **NonIdentifyingString-OrIndex**, когда кодирование должно начаться с третьего бита октета (см. также С.14). Данное значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого или того же октета.

С.15.3 При наличии альтернативы **literal-character-string** к потоку битов добавляют бит '0' (дискриминант) и **literal-character-string** кодируют, как описано в двух следующих подпунктах.

С.15.3.1 Если значение компонента **add-to-table** равно **TRUE**, то к потоку битов добавляют бит '1', в противном случает добавляют бит '0'.

С.15.3.2 Значение компонента **character-string** кодируют, как описано в С.20.

С.15.4 При наличии альтернативы **string-index** к потоку битов добавляют бит '1' (дискриминант) и **string-index** кодируют, как описано в С.28.

С.16 Кодирование типа NameSurrogate

С.16.1 Тип **NameSurrogate** определен в 7.15 следующим образом:

```
NameSurrogate ::= SEQUENCE {
     prefix-string-index INTEGER(1..one-meg) OPTIONAL,
     namespace-name-string-index INTEGER(1..one-meg) OPTIONAL,
     local-name-string-index INTEGER(1..one-meg) }
     (CONSTRAINED BY {-- prefix-string-index должен
                      -- присутствовать, только если
                      -- присутствует namespace-name-
                      -- string-index --})
```

С.16.2 Значение типа **NameSurrogate** кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда начинают с седьмого бита октета и заканчивают на восьмом бите другого октета.

С.16.3 При наличии опционального компонента **prefix-string-index** к потоку битов добавляют бит '1' (наличие), в противном случае добавляют бит '0' (отсутствие).

С.16.4 При наличии опционального компонента **namespace-name-string-index** к потоку битов добавляют бит '1' (наличие), в противном случае добавляют бит '0' (отсутствие).

С.16.5 При наличии опционального компонента **prefix-string-index** к потоку битов добавляют бит '0' (забивка) и компонент кодируют, как описано в С.25.

С.16.6 При наличии опционального компонента **namespace-name-string-index** к потоку битов добавляют бит '0' (забивка) и компонент кодируют, как описано в С.25.

С.16.7 К потоку битов добавляют бит '0' (забивка) и компонент **local-name-string-index** кодируют, как описано в С.25.

С.17 Кодирование типа QualifiedNameOrIndex, начиная со второго бита октета

С.17.1 Тип **QualifiedNameOrIndex** определен в 7.16 следующим образом:

```
QualifiedNameOrIndex ::= CHOICE {
     literal-qualified-name SEQUENCE {
          prefix IdentifyingStringOrIndex OPTIONAL
                    -- Категория PREFIX --,
          namespace-name IdentifyingStringOrIndex OPTIONAL
                    -- Категория NAMESPACE NAME --,
          local-name IdentifyingStringOrIndex
                    -- Категория LOCAL NAME -- },
     name-surrogate-index INTEGER (1..one-meg) }
```

С.17.2 В данном пункте (С.17) описано кодирование значения типа **QualifiedNameOrIndex**, когда кодирование должно начаться со второго бита октета (см. также С.18). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого или того же октета.

С.17.3 При наличии альтернативы **literal-qualified-name** к потоку битов добавляют четыре бита '1111' (идентификация) и бит '0' (забивка) и **literal-qualified-name** кодируют, как описано в четырех следующих подпунктах.

С.17.3.1 При наличии опциональных компонентов **prefix** и **namespace-name** (в указанном порядке) к потоку битов добавляют бит '1' (наличие) для каждого из них, в противном случае добавляют бит '0' (отсутствие).

С.17.3.2 При наличии опционального компонента **prefix** его кодируют, как описано в С.13.

С.17.3.3 При наличии опционального компонента **namespace-name** его кодируют, как описано в С.13.

С.17.3.4 Компонент **local-name** кодируют, как описано в С.13.

С.17.4 При наличии альтернативы **name-surrogate-index** ее кодируют, как описано в С.25.

С.18 Кодирование типа QualifiedNameOrIndex, начиная с третьего бита октета

С.18.1 Тип **QualifiedNameOrIndex** определен в 7.16 следующим образом:

```
QualifiedNameOrIndex ::= CHOICE {
     literal-qualified-name SEQUENCE {
          prefix IdentifyingStringOrIndex OPTIONAL
                    -- Категория PREFIX --,
          namespace-name IdentifyingStringOrIndex OPTIONAL
                    -- Категория NAMESPACE NAME --,
          local-name IdentifyingStringOrIndex
                    -- Категория LOCAL NAME -- },
     name-surrogate-index INTEGER (1..one-meg) }
```

С.18.2 В данном пункте (С.18) описано кодирование значения типа **QualifiedNameOrIndex**, когда кодирование должно начаться с третьего бита октета (см. также С.17). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого или того же октета.

С.18.3 При наличии альтернативы **literal-qualified-name** к потоку битов добавляют четыре бита '1111' (идентификация) и **literal-qualified-name** кодируют, как описано в четырех следующих подпунктах.

С.18.3.1 При наличии опциональных компонентов **prefix** и **namespace-name** (в указанном порядке) к потоку битов добавляют бит '1' (наличие) для каждого из них, в противном случае добавляют бит '0' (отсутствие).

С.18.3.2 При наличии опционального компонента **prefix** его кодируют, как описано в С.13.

С.18.3.3 При наличии опционального компонента **namespace-name** его кодируют, как описано в С.13.

С.18.3.4 Компонент **local-name** кодируют, как описано в С.13.

С.18.4 При наличии альтернативы **name-surrogate-index** ее кодируют, как описано в С.27.

С.19 Кодирование типа EncodedCharacterString, начиная с третьего бита октета

С.19.1 Тип **EncodedCharacterString** определен в 7.17 следующим образом:

```
EncodedCharacterString ::= SEQUENCE {
     encoding-format CHOICE {
```

```
            utf-8 NULL,
            utf-16 NULL,
            restricted-alphabet INTEGER(1..256),
            encoding-algorithm INTEGER(1..256) },
      octets NonEmptyOctetString }
```

С.19.2 В данном пункте (С.19) описано кодирование значения типа **EncodedCharacterString**, когда кодирование должно начаться с третьего бита октета (см. также С.20). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого октета.

С.19.3 Значение компонента **encoding-format** кодируют, как описано в четырех следующих подпунктах.

С.19.3.1 При наличии альтернативы **utf-8** к потоку битов добавляют два бита '00' (дискриминант).

С.19.3.2 При наличии альтернативы **utf-16** к потоку битов добавляют два бита '01' (дискриминант).

С.19.3.3 При наличии альтернативы **restricted-alphabet** к потоку битов добавляют два бита '10' (дискриминант) и **restricted-alphabet** кодируют, как описано в С.29.

С.19.3.4 При наличии альтернативы **encoding-algorithm** к потоку битов добавляют два бита '11' (дискриминант) и **encoding-algorithm** кодируют, как описано в С.29.

С.19.4 Компонент **octets** кодируют, как описано в С.23.

С.20 Кодирование типа EncodedCharacterString, начиная с пятого бита октета

С.20.1 Тип **EncodedCharacterString** определен в 7.17 следующим образом:

```
EncodedCharacterString ::= SEQUENCE {
      encoding-format CHOICE {
            utf-8 NULL,
            utf-16 NULL,
            restricted-alphabet INTEGER(1..256),
            encoding-algorithm INTEGER(1..256) },
      octets NonEmptyOctetString }
```

С.20.2 В данном пункте (С.20) описано кодирование значения типа **EncodedCharacterString**, когда кодирование должно начаться с пятого бита октета (см. также С.19). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого октета.

С.20.3 Значение компонента **encoding-format** кодируют, как описано в четырех следующих подпунктах.

С.20.3.1 При наличии альтернативы **utf-8** к потоку битов добавляют два бита '00' (дискриминант).

С.20.3.2 При наличии альтернативы **utf-16** к потоку битов добавляют два бита '01' (дискриминант).

С.20.3.3 При наличии альтернативы **restricted-alphabet** к потоку битов добавляют два бита '10' (дискриминант) и **restricted-alphabet** кодируют, как описано в С.29.

С.20.3.4 При наличии альтернативы **encoding-algorithm** к потоку битов добавляют два бита '11' (дискриминант) и **encoding-algorithm** кодируют, как описано в С.29.

С.20.4 Компонент **octets** кодируют, как описано в С.24.

С.21 Кодирование длины типа sequence-of

С.21.1 В данном пункте описывается кодирование длины типа sequence-of, который закодирован с полем длины, предшествующим элементам типа sequence-of.

Примечание — Данное кодирование всегда начинают с первого бита октета и заканчивают восьмым битого другого или того же октета.

С.21.2 Если значение находится в диапазоне от 1 до 128, то к потоку битов добавляют бит '0', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из семи битов и добавляют к потоку битов.

С.21.3 Если значение находится в диапазоне от 129 до 2^{20}, то к потоку битов добавляют бит '1' и три бита '000' (забивка), значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из двадцати битов и добавляют к потоку битов.

С.22 Кодирование типа NonEmptyOctetString, начиная со второго бита октета

С.22.1 Тип **NonEmptyOctetString** определен в 7.2 следующим образом:

```
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
```

С.22.2 В данном пункте (С.22) описано кодирование значения типа **NonEmptyOctetString**, когда кодирование должно начаться со второго бита октета (см. также С.23 и С.24). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого октета.

С.22.3 Длину строки октетов кодируют, как описано в трех следующих подпунктах.

С.22.3.1 Если длина находится в диапазоне от 1 до 64, то к потоку битов добавляют бит '0', длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из шести битов и добавляют к потоку битов.

С.22.3.2 Если длина находится в диапазоне от 65 до 320, то к потоку битов добавляют два бита '10' и пять битов '00000' (забивка), длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из восьми битов и добавляют к потоку битов.

С.22.3.3 Если длина находится в диапазоне от 321 до 2^{32}, то к потоку битов добавляют два бита '11' и пять битов '00000' (забивка), длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из тридцати двух битов и добавляют к потоку битов.

С.22.4 Биты, образующие октеты строки октетов, добавляют к потоку битов (по порядку).

С.23 Кодирование типа NonEmptyOctetString, начиная с пятого бита октета

С.23.1 Тип **NonEmptyOctetString** определен в 7.2 следующим образом:

```
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
```

С.23.2 В данном пункте (С.23) описано кодирование значения типа **NonEmptyOctetString**, когда кодирование должно начаться с пятого бита октета (см. также С.22 и С.24). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого октета.

С.23.3 Длину строки октетов кодируют, как описано в трех следующих подпунктах.

С.23.3.1 Если длина находится в диапазоне от 1 до 8, то к потоку битов добавляют бит '0', длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из трех битов и добавляют к потоку битов.

С.23.3.2 Если длина находится в диапазоне от 9 до 265, то к потоку битов добавляют два бита '10' и два бита '00' (забивка), длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из восьми битов и добавляют к потоку битов.

С.23.3.3 Если длина находится в диапазоне от 265 до 2^{32}, то к потоку битов добавляют два бита '11' и два бита '00' (забивка), длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из тридцати двух битов и добавляют к потоку битов.

С.23.4 Биты, образующие октеты строки октетов, добавляют к потоку битов (по порядку).

С.24 Кодирование типа NonEmptyOctetString, начиная с седьмого бита октета

С.24.1 Тип **NonEmptyOctetString** определен в 7.2 следующим образом:

```
NonEmptyOctetString ::= OCTET STRING (SIZE(1..four-gig))
```

С.24.2 В данном пункте (С.24) описано кодирование значения типа **NonEmptyOctetString**, когда кодирование должно начаться с седьмого бита октета (см. также С.22 и С.23). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого октета.

С.24.3 Длину строки октетов кодируют, как описано в трех следующих подпунктах.

С.24.3.1 Если длина находится в диапазоне от 1 до 2, то к потоку битов добавляют бит '0', длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из одного бита и добавляют к потоку битов.

С.24.3.2 Если длина находится в диапазоне от 3 до 258, то к потоку битов добавляют два бита '10', длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из восьми битов и добавляют к потоку битов.

С.24.3.3 Если длина находится в диапазоне от 258 до 2^{32}, то к потоку битов добавляют два бита '11', длину минус нижняя граница диапазона кодируют как целое значение без знака в поле из тридцати двух битов и добавляют к потоку битов.

С.24.4 Биты, образующие октеты строки октетов, добавляют к потоку битов (по порядку).

С.25 Кодирование целых значений в диапазоне от 1 до 2^{20}, начиная со второго бита октета

С.25.1 В данном пункте (С.25) описано кодирование целых значений в диапазоне от 1 до 2^{20}, когда кодирование должно начаться со второго бита октета (см. также С.26, С.27 и С.28). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого октета.

С.25.2 Если значение находится в диапазоне от 1 до 64, то к потоку битов добавляют бит '0', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из шести битов и добавляют к потоку битов.

С.25.3 Если значение находится в диапазоне от 65 до 8256, то к потоку битов добавляют два бита '10', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из тринадцати битов и добавляют к потоку битов.

С.25.4 Если значение находится в диапазоне от 8257 до 2^{20}, то к потоку битов добавляют два бита '10' и бит '0' (забивка), значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из двадцати битов и добавляют к потоку битов.

С.26 Кодирование целых значений в диапазоне от 0 до 2^{20}, начиная со второго бита октета

С.26.1 В данном пункте (С.26) описано кодирование целых значений в диапазоне от 0 до 2^{20}, когда кодирование должно начаться со второго бита октета (см. также С.25, С.27 и С.28). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого или этого же октета.

С.26.2 Если значение равно нулю, то к потоку битов добавляют семь битов '1111111'. В противном случае значение кодируют, как описано в С.25.

С.27 Кодирование целых значений в диапазоне от 1 до 2^{20}, начиная с третьего бита октета

С.27.1 В данном пункте (С.27) описано кодирование целых значений в диапазоне от 1 до 2^{20}, когда кодирование должно начаться с третьего бита октета (см. также С.25, С.26 и С.28). Значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого октета.

С.27.2 Если значение находится в диапазоне от 1 до 32, то к потоку битов добавляют бит '0', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из пяти битов и добавляют к потоку битов.

С.27.3 Если значение находится в диапазоне от 33 до 2080, то к потоку битов добавляют три бита '100', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из одиннадцати битов и добавляют к потоку битов.

С.27.4 Если значение находится в диапазоне от 2081 до 526368, то к потоку битов добавляют три бита '101', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из девятнадцати битов и добавляют к потоку битов.

С.27.5 Если значение находится в диапазоне от 526369 до 2^{20}, то к потоку битов добавляют три бита '110' и семь битов '0000000' (забивка), значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из двадцати битов и добавляют к потоку битов.

С.28 Кодирование целых значений в диапазоне от 1 до 2^{20}, начиная с четвертого бита октета

С.28.1 В данном пункте (С.28) описано кодирование целых значений в диапазоне от 1 до 2^{20}, когда кодирование должно начаться с четвертого бита октета (см. также С.25, С.26 и С.27). Данное значение кодируют, выполняя следующие действия (в указанном порядке).

Примечание — Кодирование данного типа всегда заканчивают на восьмом бите другого или того же октета.

С.28.2 Если значение находится в диапазоне от 1 до 16, то к потоку битов добавляют бит '0', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из четырех битов и добавляют к потоку битов.

С.28.3 Если значение находится в диапазоне от 17 до 1040, то к потоку битов добавляют три бита '100', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из десяти битов и добавляют к потоку битов.

С.28.4 Если значение находится в диапазоне от 1041 до 263184, то к потоку битов добавляют три бита '101', значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из восемнадцати битов и добавляют к потоку битов.

С.28.5 Если значение находится в диапазоне от 263185 до 2^{20}, то к потоку битов добавляют три бита '100' и шесть битов '000000' (забивка), значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из двадцати битов и добавляют к потоку битов.

С.29 Кодирование целых значений в диапазоне от 1 до 256

С.29.1 В данном пункте (С.29) описано кодирование целых значений в диапазоне 1 до 256.

Примечание — Кодирование данного типа всегда начинают с пятого или с седьмого бита октета и заканчивают на четвертом или шестом бите следующего октета соответственно.

С.29.2 Значение минус нижняя граница диапазона кодируют как целое значение без знака в поле из восьми битов и добавляют к потоку битов.

Приложение D
(справочное)

Примеры кодирования инфо-наборов XML как документов быстрого инфо-набора

D.1 Общее описание примеров

D.1.1 В настоящем приложении используют следующие типографические условные обозначения для чисел:

a) для десятичных чисел используют **полужирный шрифт Courier New** с последующим нижним индексом «10» (например, $\mathbf{11}_{10}$);

b) для шестнадцатеричных чисел используют **полужирный шрифт Courier New** с последующим нижним индексом «16» (например, $\mathbf{0b1f}_{16}$);

c) если система счисления указана явно, то нижний индекс опускают.

D.1.2 В настоящем приложении приведены два примера возможных кодирований заказа на универсальном бизнес-языке (UBL) [1] в документ быстрого инфо-набора. UBL разработан для обеспечения универсально понятного и общепризнанного коммерческого синтаксиса для юридически значимых деловых документов.

D.1.3 Инфо-набор XML для примера заказа UBL приведен в D.3.

D.1.4 Первый документ быстрого инфо-набора имеет исходный словарь, который ссылается на внешний словарь. В D.4 описано содержимое внешнего словаря, октеты документа быстрого инфо-набора и объяснены некоторые последовательности октетов.

D.1.5 Второй быстрый инфо-набор не имеет исходного словаря. В D.5 описаны октеты документа быстрого инфо-набора и объяснены некоторые последовательности октетов.

Примечание — Окончательный словарь последнего документа быстрого инфо-набора тот же самый, что и окончательный словарь документа быстрого инфо-набора, описанного в D.4.

D.1.6 В D.4 и D.5 октеты представлены рядом таблиц по две графы в каждой. В первой графе перечислены в шестнадцатеричном виде начальные позиции 32 последовательных октетов документа быстрого инфо-набора, а во второй графе приведены эти октеты в шестнадцатеричной нотации. Шестнадцатеричные символы, содержащие биты, которые соответствуют идентификации и указателю конца информационных элементов, подчеркнуты.

D.1.7 Объяснения некоторых последовательностей октетов документов быстрого инфо-набора (в D.4 и D.5) представлены в таблицах в следующих графах:

a) графа 1 содержит в шестнадцатеричном виде позицию октета(ов), приведенного(ых) в графе 2;

b) графа 2 содержит октет(ы) документа быстрого инфо-набора, связанный(е) с соответствующим информационным элементом и его свойствами. Октет представлен в двоичном виде с последующим заключенным в скобки шестнадцатеричным представлением [например, **11110000 (f0)**];

c) графа 3 содержит подробное описание приведенного в графе 2 октета и ссылки на пункты приложения С для дальнейших объяснений и уточнений;

d) графа 4 содержит часть инфо-набора XML или документа XML 1.0 (если применимо), соответствующую октету(ам), приведенному(ым) в графе 2.

D.1.8 В приведенных примерах все блоки информационных элементов **character**, содержащие менее 6 символов, добавлены в таблицу CONTENT CHARACTER CHUNK, а значения свойства **[normalized value]** всех информационных элементов **attribute**, содержащие менее 6 символов, добавлены в таблицу ATTRIBUTE VALUE.

D.1.9 Размеры документов XML 1.0 и быстрого инфо-набора, а также размеры этих сжатых (с помощью GZIP) документов приведены в D.2.

D.2 Размер документов из примеров (включая сжатые на основании избыточности)

D.2.1 В таблице D.1 приведены размеры всех документов. В графе 1 приведены документы UBL, в графе 2 — размеры документов, в графе 3 — размеры документов, сжатых с помощью GZIP (с принятыми по умолчанию опциями) [2].

Примечания

1 Документ XML 1.0 UBL Order не содержит пробельных символов (см. D.3.1.2).

2 В каждом документе все символы закодированы с использованием кодирования символов UTF-8.

3 Для документов быстрого инфо-набора не сериализована декларация XML (см. 12.3).

Таблица D.1 — Исходные размеры документов и размеры документов после сжатия с помощью GZIP

Документ UBL	Размер	Размер после сжатия GZIP
Документ XML 1.0	3311	893
Документ быстрого инфо-набора с внешним словарем	684	546
Документ быстрого инфо-набора без исходного словаря	1322	860

D.2.2 Размер документа быстрого инфо-набора со ссылкой на внешний словарь является наименьшим, также как и его размер после сжатия с помощью GZIP. Отношение размера сжатого GZIP документа к исходному размеру для документа быстрого инфо-набора указывает, что этот документ быстрого инфо-набора содержит мало избыточной информации.

D.2.3 Во всех случаях размер сжатых GZIP документов быстрого инфо-набора меньше размера сжатых GZIP документов XML 1.0. Более того, размер документа быстрого инфо-набора со ссылкой на внешний словарь меньше размера сжатого GZIP документа XML 1.0.

D.3 Пример UBL Order

D.3.1 Пример Joinery Order

D.3.1.1 Пример порядка UBL взят из [1]. В частности, пример Joinery Order был выбран (см. xml/joinery/UBL-Order-1.0-Joinery-Example.xml) по следующим причинам:

a) он является примером из реального мира и разработан независимо от настоящего стандарта без учета быстрого инфо-набора;

b) он находится в свободном доступе;

c) в нем широко использованы пространства имен XML, и следовательно, это хороший пример представления того, как быстрый инфо-набор поддерживает пространства имен XML.

D.3.1.2 Пример Joinery Order был изменен следующим образом:

a) последние три элемента **OrderLine** были удалены.

Примечание — Тем самым документ XML 1.0 был уменьшен до разумных размеров для представления в настоящем стандарте;

b) все пробельные символы были удалены.

Примечание — Это представляет более реалистичный случай использования инфо-наборов XML, которые могут быть сериализованы, переданы по сети и проанализированы.

D.3.2 Документ XML 1.0 Joinery Order

Ниже приведен документ XML 1.0 Joinery Order с указанными в D.3.1.2, перечисление a) изменениями, но с сохранением пробельных символов для удобства чтения:

```
<?xml version=«1.0» encoding=«UTF-8»?>
<Order xmlns:res=«urn:oasis:names:tc:ubl:codelist:AcknowledgementResponseCode:1:0»
xmlns:cbc=«urn:oasis:names:tc:ubl:CommonBasicComponents:1:0»
xmlns:cac=«urn:oasis:names:tc:ubl:CommonAggregateComponents:1:0»
xmlns:cur=«urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0»
xmlns:xsi=«
xmlns=«urn:oasis:names:tc:ubl:Order:1:0»
xsi:schemaLocation=«urn:oasis:names:tc:ubl:Order:1:0 ../../xsd/maindoc/UBL-Order-1.0.xsd»>
    <BuyersID>S03-034257</BuyersID>
    <cbc:IssueDate>2003-02-03</cbc:IssueDate>
    <cac:BuyerParty>
        <cac:Party>
            <cac:PartyName>
                <cbc:Name>Jerry Builder plc</cbc:Name>
            </cac:PartyName>
            <cac:Address>
```

gost.e-ecolog.ru

```
                <cbc:StreetName>Marsh Lane</cbc:StreetName>
                <cbc:CityName>Nowhere</cbc:CityName>
                <cbc:PostalZone>NR18 4XX</cbc:PostalZone>
                <cbc:CountrySubentity>Norfolk</cbc:CountrySubentity>
            </cac:Address>
            <cac:Contact>
                <cbc:Name>Eva Brick</cbc:Name>
            </cac:Contact>
        </cac:Party>
    </cac:BuyerParty>
    <cac:SellerParty>
        <cac:Party>
            <cac:PartyName>
                <cbc:Name>Specialist Windows plc</cbc:Name>
            </cac:PartyName>
            <cac:Address>
                <cbc:BuildingName>Snowhill Works</cbc:BuildingName>
                <cbc:CityName>Little Snoring</cbc:CityName>
                <cbc:PostalZone>SM2 3NW</cbc:PostalZone>
                <cbc:CountrySubentity>Whereshire</cbc:CountrySubentity>
            </cac:Address>
        </cac:Party>
    </cac:SellerParty>
    <cac:Delivery>
        <cbc:RequestedDeliveryDateTime>2003-02-24T00:00:00
        </cbc:RequestedDeliveryDateTime>
        <cac:DeliveryAddress>
            <cbc:StreetName>Riverside Rd.</cbc:StreetName>
           <cbc:BuildingName>Plot 17, Whitewater Estate</cbc:BuildingName>
            <cbc:CityName>Whetstone</cbc:CityName>
            <cbc:CountrySubentity>Middlesex</cbc:CountrySubentity>
        </cac:DeliveryAddress>
    </cac:Delivery>
    <cac:OrderLine>
        <cac:LineItem>
            <cac:BuyersID>A</cac:BuyersID>
            <cbc:Quantity quantityUnitCode=«unit»>2</cbc:Quantity>
            <cac:Item>
                <cac:SellersItemIdentification>
                    <cac:ID>236WV</cac:ID>
                    <cac:PhysicalAttribute>
                        <cac:AttributeID>wood</cac:AttributeID>
                        <cbc:Description>soft</cbc:Description>
                    </cac:PhysicalAttribute>
                    <cac:PhysicalAttribute>
                        <cac:AttributeID>finish</cac:AttributeID>
                        <cbc:Description>primed</cbc:Description>
                    </cac:PhysicalAttribute>
                    <cac:PhysicalAttribute>
                        <cac:AttributeID>fittings</cac:AttributeID>
                        <cbc:Description>satin</cbc:Description>
                    </cac:PhysicalAttribute>
                    <cac:PhysicalAttribute>
                        <cac:AttributeID>glazing</cac:AttributeID>
                        <cbc:Description>single</cbc:Description>
                    </cac:PhysicalAttribute>
```

```
                    </cac:SellersItemIdentification>
               </cac:Item>
          </cac:LineItem>
     </cac:OrderLine>
     <cac:OrderLine>
          <cac:LineItem>
               <cac:BuyersID>B</cac:BuyersID>
               <cbc:Quantity quantityUnitCode=«unit»>3</cbc:Quantity>
               <cac:Item>
                    <cac:SellersItemIdentification>
                         <cac:ID>340TW</cac:ID>
                         <cac:PhysicalAttribute>
                              <cac:AttributeID>hand</cac:AttributeID>
                              <cbc:Description>RH</cbc:Description>
                         </cac:PhysicalAttribute>
                         <cac:PhysicalAttribute>
                              <cac:AttributeID>wood</cac:AttributeID>
                              <cbc:Description>hard</cbc:Description>
                         </cac:PhysicalAttribute>
                         <cac:PhysicalAttribute>
                              <cac:AttributeID>finish</cac:AttributeID>
                              <cbc:Description>stain</cbc:Description>
                         </cac:PhysicalAttribute>
                         <cac:PhysicalAttribute>
                              <cac:AttributeID>fittings</cac:AttributeID>
                              <cbc:Description>brass</cbc:Description>
                         </cac:PhysicalAttribute>
                         <cac:PhysicalAttribute>
                              <cac:AttributeID>glazing</cac:AttributeID>
                              <cbc:Description>double</cbc:Description>
                         </cac:PhysicalAttribute>
                    </cac:SellersItemIdentification>
               </cac:Item>
          </cac:LineItem>
     </cac:OrderLine>
</Order>
```

D.4 Документ быстрого инфо-набора UBL Order с внешним словарем

Внешний словарь документа быстрого инфо-набора приведен в D.4.1. Октеты (как шестнадцатеричные символы) приведены в D.4.2. Подробные объяснения некоторых последовательностей октетов в D.4.2 приведены в D.4.3. Документ быстрого инфо-набора не может рассматриваться как самоописывающийся, так как для создания полного инфо-набора XML требуется внешняя информация (внешний словарь).

Примечание — Документ быстрого инфо-набора может быть обработан синтаксическим анализатором быстрого инфо-набора, который не может получить словарные таблицы, заданные URI, но индексы словарных таблиц не будут перенаправлены для получения информации, необходимой для генерации свойств информационных элементов.

D.4.1 Внешний словарь UBL Order

D.4.1.1 Внешний словарь документа быстрого инфо-набора по определению должен быть окончательным словарем, полученным из инфо-набора XML примера UBL Order (см. D.3.1.2) и после измененным для того, чтобы:

a) не содержать информационных элементов **character**;

b) содержать пустые свойства **[normalized value]** информационных элементов **attribute**.

Примечания

1 Эти изменения представляют реалистичный сценарий, когда заранее неизвестно, каким будет определяемое приложением содержимое (информационные элементы **character** и/или свойства **[normalized value]** информационных элементов **attribute**) инфо-набора XML.

2 На практике не ожидается, что подлежащий сериализации документ будет использован для создания внешнего словаря. Предполагается, что реализующие средства будут использовать схему и потенциально экземпляры инфо-набора XML схемы в результате частотного анализа строк и квалифицированных имен станут такими, что меньшие значения индексов будут присвоены более часто появляющейся информации (например, частота появления свойств **[local name]** в инфо-наборах XML может быть описана степенным рядом).

D.4.1.2 URI внешнего словаря: **urn:oasis:names:tc:ubl:Order:1.0:joinery:example**.

D.4.1.3 В таблице D.2 представлен словарь инфо-набора XML UBL Order (словарные таблицы). В графе 1 перечислены индексы словарных таблиц (индекс), в графе 2 — записи словарной таблицы PREFIX (запись префикса), в графе 3 — записи словарной таблицы NAMESPACE NAME (запись имени пространства имен), в графе 4 — записи словарной таблицы LOCAL NAME (запись локального имени), в графе 5 — записи словарной таблицы ELEMENT NAME (запись имени элемента), в графе 6 — записи словарной таблицы ATTRIBUTE NAME (запись имени атрибута). Значения индексов для записей идентификаторов имен таблиц ELEMENT NAME и ATTRIBUTE NAME представлены в порядке, определенном для компонентов типа **`NameSurrogate`** (**`prefix-name-string-index`**, **`namespace-name-string-index`** и **`local-name-string-index`**). Символ «_» указывает на отсутствие значения (что возможно только для значений компонентов **`prefix-name-string-index`** и **`namespace-name-string-index`**).

Примечания

1 Первая запись (индекс 1) для префикса и имени пространства имен, соответствующая префиксу XML («xml») и имени пространства имен XML («http://www.w3.org/XML/1998/namespace»), является встроенной (см. 7.2.21 и 7.2.22).

2 Длинные записи имен пространств имен (URI) были урезаны.

3 Для первой записи имени элемента (индекс 1) нет ссылки на префикс (так как значение отсутствует, на что указывает «_»), есть ссылки на седьмую запись имени пространства имен (индекс 7) для свойства **[namespace name]** («urn:oasis:names:tc:ubl:Order:1:0») и на первую запись локального имени (индекс 1) для свойства **[local name]** («Order»).

Таблица D.2 — Словарь инфо-набора XML UBL Order

Индекс	Запись префикса	Запись имени пространства имен	Запись локального имени	Запись имени элемента	Запись имени атрибута
1	xml	gost.e-ecolog.ru 1998/namespace	Order	_ 7 1	6 6 2
2	res	....AcknowledgementResponseCode:1:0	schemaLocation	_ 7 3	_ _ 23
3	cbc	....CommonBasicComponents:1:0	BuyersID	3 3 4	
4	cac	....CommonAggregateComponents:1:0	IssueDate	4 4 5	
5	cur	....CurrencyCode:1:0	BuyerParty	4 4 6	
6	xsi	....XMLSchema-instance	Party	4 4 7	
7		....Order:1:0	PartyName	3 3 8	
8			Name	4 4 9	
9			Address	3 3 10	
10			StreetName	3 3 11	
11			CityName	3 3 12	
12			PostalZone	3 3 13	
13			CountrySubentity	4 4 14	
14			Contact	4 4 15	
15			SellerParty	3 3 16	
16			BuildingName	4 4 17	
17			Delivery	3 3 18	
18			RequestedDeliveryDate-Time	4 4 19	
19			DeliveryAddress	4 4 20	
20			OrderLine	4 4 21	
21			LineItem	4 4 3	
22			Quantity	3 3 22	
23			quantityUnitCode	4 4 24	
24			Item	4 4 25	
25			SellersItemIdentification	4 4 26	
26			ID	4 4 27	
27			PhysicalAttribute	4 4 28	
28			AttributeID	3 3 29	
29			Description		

D.4.2 Октеты документа быстрого инфо-набора (в виде шестнадцатеричных символов)

В таблице D.3 приведены октеты документа быстрого инфо-набора для примера UBL Order, представленного в D.3.

Примечание — В таблице D.3 подчеркнуты шестнадцатеричные символы, содержащие биты, которые соответствуют идентификации и указателю конца информационных элементов.

Таблица D.3 — Октеты документа быстрого инфо-набора (в виде шестнадцатеричных символов)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e001000020100 02f75726e3a6f617369733a6e616d65733a74633a75626c3a4f
000020	726465723a313a303a6a6f696e6572793a65786164706c6578cf8181cf8282cf
000040	8383cf8484cf8585cd86f00000083b75726e3a6f617369733a6e616d65733a74
000060	633a75626c3a4f726465723a313a30202e2e2f2e2e2f7873642f6d61696e646f
000080	632f55424c2d4f726465722d312e302e787364f00182075330332d3033343235
0000a0	37f00282073230303332d30322d3033f003040506820e4a6572727920427 5696c
0000c0	6465722070 6c63ff07088207 4d6172736820 4c616e65f00982044e6f77686572
0000e0	65f00a82054e523138203458 58f00b82044e6f72666f6c6bff0c068206457661
000100	20427269636bffff0d040506821353706563696 16c6973742057696e646f7773
000120	20706c63ff070e820b536e6f7768696c6c20576f726b73f009820b4c6974746c
000140	6520536e6f72696e67f00a8204534d3220334e57f00b8207576865726 5736869
000160	7265ffff0f1082103230303332d30322d32345430303a30303a3030f01108820a
000180	52697665727369646520526 42ef00e821750 6c6f742031372c20576869746577
0001a0	617465722045737461746 5f009820657686574737467 46f6e65f00b82064d696464
0001c0	6c6573657 8fff012131490 41f055014375 6e6974f09032f016171892023233 36
0001e0	5756f0191a9201776f6f64f01b9201736f6674ff191a820366696e697368f01b
000200	8203707269 6d6564ff191a8205666974746 96e6773f01b9202736174696eff19
000220	1a8204676c617a696e67f01b82037369 6e676c65ffffff213149042f0550180
000240	f09033f016171892023334305457f0191a9201686 16e64f01b91524 8ff191aa3
000260	f01b9201686172 64ff191a8203666696e697368f01b9202737461696eff191a82
000280	05666974746 96e6773f01b9202627261737 3ff191a8204676c617a696e67f01b
0002a0	8203646f75626c65ffffffff
0002ac	

D.4.3 Объяснение кодирования

D.4.3.1 Кодирование информационных элементов document и Order element

Ниже приведено объяснение деталей исходного кодирования документа быстрого инфо-набора (включая URI внешнего словаря) и корневого информационного элемента. В частности, объяснено кодирование информационного элемента **document**, последовательности информационных элементов **namespace**, информационного элемента **element** и информационного элемента **attribute**. В таблице D.4 приведен фрагмент документа быстрого инфо-набора для кодирования информационных элементов **document** и **Order element** из D.3.2. В таблице D.5 данное кодирование подробно описано. Данный фрагмент в XML 1.0 представлен следующим образом:

```
<Order xmlns:res=«urn:oasis:names:tc:ubl:codelist:AcknowledgementResponseCo
de:1:0»
    xmlns:cbc=«urn:oasis:names:tc:ubl:CommonBasicComponents:1:0»
    xmlns:cac=«urn:oasis:names:tc:ubl:CommonAggregateComponents:1:0»
    xmlns:cur=«urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0»
    xmlns:xsi=«
    xmlns=«urn:oasis:names:tc:ubl:Order:1:0»
    xsi:schemaLocation=«urn:oasis:names:tc:ubl:Order:1:0  ../../xsd/maindoc/UBL-
Order-1.0.xsd»>
```

gost.e-ecolog.ru

Таблица D.4 — Октеты фрагмента (в виде шестнадцатеричных символов)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e0010000201000 2f75726e3a6f617369733a6e616d65733a74633a75626c3a4f
000020	726465723a313a303a6a6f696e6572793a6578616d706c6578cf8181cf8282cf
000040	8383cf8484cf8585cd86f0000083b75726e3a6f617369733a6e616d65733a74
000060	633a75626c3a4f726465723a313a30202e2e2f2e2e2f7873642f6d61696e646f
000080	632f55424c2d4f726465722d312e302e787364f0

Таблица D.5 — Детали кодирования

	Октет(ы)	Описание	Инфо-набор XML или XML
00 01	11100000 (e0) 00000000 (00)	Эти октеты присутствуют в начале каждого документа быстрого инфо-набора (см. 12.6)	Информационный элемент **document**
02 03	00000000 (00) 00000001 (01)	Эти октеты являются кодированием номера версии (см. 12.9)	
04 05 06	00100000 (20) 00010000 (10) 00000000 (00)	Эти октеты являются кодированием наличия исходного словаря и ссылки на внешний словарь из исходного словаря. Октет на позиции 04_{16} со значением 20_{16} имеет значение '0' (забивка) в первом бите (см. 12.8). Третий бит равен '1', обозначая наличие компонента **initial-vocabulary** и отсутствие остальных шести опциональных компонентов (см. C.2.3). Октет на позиции 05_{16} со значением 10_{16} имеет в трех первых битах '0' (забивка) (см. C.2.5). Четвертый бит равен '1', обозначая наличие компонента **external-vocabulary** из **initial-vocabulary**. Последние четыре бита (с пятого по восьмой) равны '0', обозначая отсутствие четырех из двенадцати других опциональных компонентов (см. C.2.5.1). Октет на позиции 06_{16} со значением 00_{16} имеет значение '0' во всех битах, обозначая отсутствие оставшихся восьми из двенадцати опциональных компонентов (см. C.2.5.1)	

Продолжение таблицы D.5

	Октет(ы)	Описание	Инфо-набор XML или XML
07 08 37	00101111 (2f) 01110101 (75) 01100101 (65)	Эти октеты являются кодированием URI внешнего словаря. Октет на позиции 07_{16} со значением $2f_{16}$ имеет значение '0' (забивка) в первом бите (см. С.2.5.2). URI закодирован как символы UTF-8 (см. С.22). Второй бит равен '0', обозначая, что длина URI больше или равна 1_{10} октету и меньше или равна 64_{10} октетам и длина минус нижняя граница закодирована в битах с третьего по восьмой как целое без знака (см. С.22.3.1). Целое без знака равно 47_{10}, и длина равна 48_{10} (нижняя граница равна 1_{10}). 48_{10} октетов символов UTF-8 (URI) закодированы, начиная с октета на позиции 08_{16} до октета на позиции 37_{16}	
38	01111000 (78)	Данный октет является исходным кодированием дочернего элемента информационного элемента **document**. Октет на позиции 38_{16} со значением 78_{16} имеет значение '0' (идентификация) для первого бита, обозначая, что у информационного элемента **document** есть дочерний элемент и им является информационный элемент **element** (см. С.2.11.2). Второй бит равен '1', обозначая, что информационный элемент **element** имеет атрибуты (см. С.3.3). Биты с третьего по шестой равны '1110' с последующими '00' (забивка) в битах с седьмого по восьмой, обозначая наличие пространства имен информационного элемента **attribute** (см. С.3.4.1)	Информационный элемент **element** со свойством **[namespace attribute]**
39 3a 3b	11001111 (cf) 10000001 (81) 10000001 (81)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с индексированными свойствами **[prefix]** и **[normalized value]**. Октет на позиции 39_{16} со значением cf_{16} имеет значение '110011' (идентификация) в битах с первого по шестой, обозначая наличие пространства имен информационного элемента **attribute** (см. С.3.4.2). Седьмой бит равен '1', обозначая наличие свойства **[prefix]**. Восьмой бит равен '1', обозначая наличие свойства **[normalized value]**. Октет на позиции $3a_{16}$ со значением 81_{16} имеет '1' в первом бите, обозначая, что индекс закодирован и будет идентифицировать свойство **[prefix]** в таблице PREFIX (см. С.13.4). Второй бит равен '0', обозначая, что индекс больше или равен 1_{10} и меньше или равен 64_{10} и закодирован в битах с третьего по восьмой как целое без знака (см. С.25.2). Целое без знака равно 1_{10}, и индекс равен 2_{10} (нижняя граница равна 1_{10}), что после перенаправления из таблицы PREFIX приводит к значению свойства **[prefix]**, равному «res». Октет на позиции $3b_{16}$ со значением 81_{16} имеет '1' в первом бите, обозначая, что индекс закодирован и идентифицирует свойство **[normalized value]** в таблице NAMESPACE NAME (см. С.13.4). Второй бит равен '0', обозначая, что индекс больше или равен 1_{10} и меньше или равен 64_{10} и закодирован в битах с третьего по восьмой как целое без знака (см. С.25.2). Целое без знака равно 1_{10}, и индекс равен 2_{10} (нижняя граница равна 1_{10}), что после перенаправления из таблицы NAMESPACE NAME приводит к значению свойства **[normalized value]**, равному «.ResponseCode:1.0»	**xmlns:res= «...ResponseCode:1:0»**
3c 3d 3e	11001111 (cf) 10000010 (82) 10000010 (82)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с индексированными свойствами **[prefix]** и **[normalized value]**. Индекс свойства **[prefix]** равен 3_{10}, что после перенаправления из таблицы PREFIX дает значение «cbc». Индекс свойства **[normalized value]** равен 3_{10}, что после перенаправления из таблицы NAMESPACE NAME дает значение «...sicComponents:1.0»	**xmlns:cbc= «..sicComponents:1:0»**

Продолжение таблицы D.5

	Октет(ы)	Описание	Инфо-набор XML или XML
3f 40 41	11001111 (cf) 10000011 (83) 10000011 (83)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с индексированными свойствами **[prefix]** и **[normalized value]**	**xmlns:cac= «...ateComponents:1:0»**
42 43 44	11001111 (cf) 10000100 (84) 10000100 (84)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с индексированными свойствами **[prefix]** и **[normalized value]**	**xmlns:cur= «...CurrencyCode:1:0»**
45 46 47	11001111 (cf) 10000101 (85) 10000101 (85)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с индексированными свойствами **[prefix]** и **[normalized value]**	**xmlns:xsi= «...Schema-instance»**
48 49	11001101 (cd) 10000110 (86)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с индексированным свойством **[normalized value]**. Октет на позиции 48_{16} со значением cd_{16} имеет седьмой бит, равный '0', обозначающий, что свойство **[prefix]** отсутствует, восьмой бит, равный '1', означает, что свойство **[normalized value]** присутствует	**xmlns=«...Order:1:0»**
4a	11110000 (f0)	Этот октет является кодированием указателя конца для последовательности пространств имен информационного элемента **attribute**. Октет на позиции $4a_{16}$ со значением $f0_{16}$ имеет значение '1111' (указатель конца) в первых четырех битах (с первого по четвертый), и эти биты являются указателем конца последовательности. Четыре из шести значений '0' (забивка) содержатся в битах с пятого по восьмой (см. С.3.4.3)	
4b	00000000 (00)	Этот октет является кодированием индексированного квалифицированного имени информационного элемента **element**. Октет на позиции $4b_{16}$ со значением 00_{16} содержит два оставшихся из шести '0' (забивка) в первом и втором битах (см. С.3.4.3). Третий бит равен '0', обозначая, что квалифицированное имя является не литеральным квалифицированным именем (см. С.18.3), а индексированным. Индекс больше или равен 1_{10} и меньше или равен 32_{10} и закодирован в битах с четвертого по восьмой как целое без знака (см .С.27.2). Целое без знака равно 0_{10}, и индекс равен 1_{10} (нижняя граница равна 1_{10}), что после перенаправления из таблицы ELEMENT NAME дает квалифицированное имя со свойством **[namespace name]**, равным «....Order:1.0» и свойством **[local name]** равным «Order» (для этого квалифицированного имени нет свойства **[prefix]**)	**<Order ....**
4c 4d 4e 4f 92	00000000 (00) 00001000 (08) 01111011 (3b) 01110101 (75) 01101000 (64)	Эти октеты являются кодированием информационного элемента **attribute** с индексированным квалифицированным именем и со свойством **[normalized value]**. Наличие информационных элементов **attribute** было обозначено в октете позиции 38_{16} (второй бит равен '1'). Октет на позиции $4c_{16}$ со значением 00_{16} имеет первый бит '0' (идентификация), обозначая наличие информационного элемента **attribute** (см. С.3.6.1). Второй бит равен '0', обозначая, что квалифицированное имя является не литеральным (см. С.17.3), а индексированным. Индекс больше или равен 1_{10} и меньше или равен 64_{10} и закодирован в битах с третьего по восьмой как целое без знака (см. С.25.2). Целое без знака равно 0_{10}, и индекс равен 1_{10} (нижняя граница равна 1_{10}), что после перенаправления из таблицы ATTRIBUTE NAME дает квалифицированное имя с равным «xsi» свойством **[prefix]**, равным «....Schema-instance» свойством **[namespace name]** и равным «schemaLocation» свойством **[local name]**	**xsi:schemaLocation= «...»**

Окончание таблицы D.5

	Октет(ы)	Описание	Инфо-набор XML или XML
		Октет на позиции $4d_{16}$ со значением 08_{16} является исходным кодированием неидентифицирующей строки или индекса (см. С.14) для свойства **[normalized value]**. Первый бит равен '0', обозначая наличие литеральной строки символов (см. С.14.3). Второй бит равен '0', обозначая, что литеральную строку символов не следует добавлять в таблицу ATTRIBUTE VALUE. Третий и четвертый биты равны '0', обозначая, что форматом кодирования строки является UTF-8 (см. С.19.3.1). Пятый и шестой биты равны '1' и '0' соответственно, обозначая, что длина октетов закодированных символов UTF-8 (свойство **[normalized value]**) больше или равна 9_{10} и меньше или равна 264_{10} октетам и длина минус нижняя граница закодирована в восьми битах следующего октета как целое без знака (см. С.23.3.2). Седьмой и восьмой биты равны '0' (забивка) (см. С.23.3.2). Октет на позиции $4e_{16}$ со значением $3b_{16}$ является кодированием целого без знака. Длина в октетах закодированных символов UTF-8 равна 68_{10} (нижняя граница равна 9_{10}). 68_{10} октетов закодированных символов UTF-8 (свойства **[normalized value]**) являются кодированием с октета на позиции $4f_{16}$ до октета на позиции 92_{16}	
93	11110000 (f0)	Этот октет является кодированием указателя конца последовательности информационных элементов **attribute**. Октет на позиции 93_{16} со значением $f0_{16}$ содержит '1111' в первых четырех битах (с первого по четвертый бит), что является указателем конца последовательности. Четыре '0' (забивка) присутствуют (с пятого по восьмой бит), так как информационный элемент **Order element** имеет дочерние элементы (см. D.3.2)	

D.4.3.2 Кодирование информационного элемента Address element из информационного элемента BuyerParty element

Ниже приведено объяснение деталей кодирования информационного элемента **Address element** из информационного элемента **BuyerParty element** документа быстрого инфо-набора. В частности, объяснено кодирование информационных элементов **element** и **character**. В таблице D.6 приведен фрагмент документа быстрого инфо-набора для кодирования информационного элемента **Address element** из информационного элемента **BuyerParty element** (D.3.2). В таблице D.7 описаны детали этого кодирования. Такой фрагмент в XML 1.0 выглядит следующим образом:

```
<cac:Address>
        <cbc:StreetName>Marsh Lane</cbc:StreetName>
        <cbc:CityName>Nowhere</cbc:CityName>
        <cbc:PostalZone>NR18 4XX</cbc:PostalZone>
<cbc:CountrySubentity>Norfolk</cbc:CountrySubentity>
</cac:Address>
```

Таблица D.6 — Октеты фрагмента (в виде шестнадцатеричных символов)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
0000c0	07088207 4d617273 68204c61 6e65f009 82044e6f 77686572
0000e0	65f00a82 054e5231 38203458 58f00b82 044e6f72 666f6cbff

Таблица D.7 — Детали кодирования

	Октет(ы)	Описание	Инфо-набор XML или XML
c8	00000111 (07)	Этот октет является кодированием информационного элемента **Address element**. Октет на позиции $c8_{16}$ со значением 07_{16} имеет '0' (идентификация) в первом бите, обозначая, что у информационного элемента **element** есть дочерний элемент (дочерний информационного элемента **Party element**), который является информационным элементом **element** (см. С.3.7.2). Второй бит равен '0', обозначая, что у информационного элемента **element** нет атрибутов (см. С.3.3). Третий бит равен '0', обозначая, что квалифицированное имя является не литеральным (см. С.18.3), а индексированным. Индекс больше или равен 1_{10} и меньше или равен 32_{10} и закодирован в битах с четвертого по восьмой как целое без знака (см. С.27.2). Целое без знака равно 7_{10}, и индекс равен 8_{10} (нижняя граница равна 1_{10}), что после перенаправления из таблицы ELEMENT NAME дает квалифицированное имя с равным «cac» свойством **[prefix]**, равным «….gateComponents:1.0» свойством **[namespace name]** и равным «Address» свойством **[local name]**	**<cac:Address>**
c9	00001000 (08)	Этот октет является кодированием информационного элемента **StreetName element**. Информационный элемент **element** имеет индекс 9_{10}, что после перенаправления из таблицы ELEMENT NAME дает квалифицированное имя с равным «cbc» свойством **[prefix]**, равным «….BasicComponents:1:0» свойством **[namespace name]** и равным «StreetName» свойством **[local name]**	**<cbc:Street-Name>**
ca cb cc d5	10000010 (82) 00000111 (07) 01001101 (4d) 01100101 (65)	Эти октеты являются кодированием информационного элемента **character** из информационного элемента **StreetName element**. Октет на позиции ca_{16} со значением 82_{16} имеет '10' (идентификация) в первых двух битах (с первого по второй биты), обозначая, что у информационного элемента **element** есть дочерний элемент (дочерний информационного элемента **StreetName element**), который является блоком информационных элементов **character** (см. С.3.7.5). Третий бит равен '0', обозначая наличие литеральной строки символов (см. С.15.3). Четвертый бит равен '0', обозначая, что литеральную строку символов не следует добавлять в таблицу CONTENT CHARACTER CHUNK. Пятый и шестой биты равны '0', обозначая, что форматом кодирования блока является UTF-8 (см. С.20.3.1). Седьмой и восьмой биты равны '1' и '0' соответственно, обозначая, что длина октетов закодированных символов UTF-8 (блока информационных элементов **character**) больше или равна 3_{10} и меньше или равна 258_{10} октетам и длина минус нижняя граница закодирована в восьми битах следующего октета как целое без знака (см. С.23.3.2). Октет на позиции cb_{16} со значением 07_{16} является кодированием целого без знака. Длина в октетах закодированных символов UTF-8 равна 10_{10} (нижняя граница равна 3_{10}). 10_{10} октетов закодированных символов UTF-8 расположены с октета на позиции cc_{16} до октета на позиции $d5_{16}$	Информационные элементы **character** «Marsh Lane»
d6	1111<u>0000</u> (f0)	Этот октет является указателем конца информационного элемента **StreetName element**. Октет на позиции $d6_{16}$ со значением $f0_{16}$ содержит '1111' в первых четырех битах (с первого по четвертый бит), что является указателем конца информационного элемента **StreetName element**. Четыре '0' (забивка) присутствуют (с пятого по восьмой бит), так как есть еще дочерний элемент (того же уровня) (информационный элемент **CityName element**) (см. С.3.7.1)	**</ cbc:StreetName>**

Окончание таблицы D.7

	Октет(ы)	Описание	Инфо-набор XML или XML
d7	00001001 (09)	Этот октет является кодированием информационного элемента **CityName element**. Информационный элемент **element** имеет индекс 10_{10}, что после перенаправления из таблицы ELEMENT NAME дает квалифицированное имя с равным «cbc» свойством **[prefix]**, равным «....BasicComponents:1:0» свойством **[namespace name]** и равным «CityName» свойством **[local name]**	**<cbc:CityName>**
d8 d9 da e0	10000010 (82) 00000100 (04) 01001110 (4e) 01100101 (65)	Эти октеты являются кодированием информационных элементов **character** из информационного элемента **CityName element**. 7_{10} октетов закодированных символов UTF-8 начинаются с октета на позиции da_{16} до октета на позиции $e0_{16}$	Информационные элементы **character** «Nowhere»
e1	11110000 (f0)	Этот октет является указателем конца информационного элемента **CityName element**	**</cbc:CityName>**
e2	00001010 (0a)	Этот октет является кодированием информационного элемента **PostalZone element**. Информационный элемент **element** имеет индекс 11_{10}, что после перенаправления из таблицы ELEMENT NAME дает квалифицированное имя с равным «cbc» свойством **[prefix]**, равным «....BasicComponents:1:0» свойством **[namespace name]** и равным «PostalZone» свойством **[local name]**	**<cbc:PostalZone>**
e3 e4 e5 .... ec	10000010 (82) 00000101 (05) 01001110 (4e) 01011000 (58)	Эти октеты являются кодированием информационных элементов **character** из информационного элемента **PostalZone element**. 8_{10} октетов закодированных символов UTF-8 начинаются с октета на позиции $e5_{16}$ до октета на позиции ec_{16}	Информационные элементы **character** «NR18 4XX»
ed	11110000 (f0)	Этот октет является указателем конца информационного элемента **PostalZone element**	**</ cbc:PostalZone>**
ee	00001011 (0b)	Этот октет является кодированием информационного элемента **CountrySubentity element**. Информационный элемент **element** имеет индекс 12_{10}, что после перенаправления из таблицы ELEMENT NAME дает квалифицированное имя с равным «cbc» свойством **[prefix]**, равным «....BasicComponents:1:0» свойством **[namespace name]** и равным «CountrySubentity» свойством **[local name]**	<cbc:CountrySubentity>
ef f0 f1 .... f7	10000010 (82) 00000100 (04) 01001110 (4e) 01101011 (6b)	Эти октеты являются кодированием информационных элементов **character** из информационного элемента **CountrySubentity element**. 7_{10} октетов закодированных символов UTF-8 начинаются с октета на позиции $f1_{16}$ до октета на позиции $f7_{16}$	Информационные элементы **character** «Norfolk»
f8	11111111 (ff)	Этот октет является указателем конца информационных элементов **CountrySubentity element** и **Address element**. Октет на позиции $f8_{16}$ со значением ff_{16} содержит '1111' в первых четырех битах (с первого по четвертый бит), что является указателем конца информационного элемента **CountrySubentity element** (см. С.3.8). Последние четыре бита (биты с пятого по восьмой) содержат '1111' и являются указателем конца информационного элемента **Address element** (см. С.3.8)	**</cbc:Country-Subentity></cac:Address>**

D.5 Документ быстрого инфо-набора UBL Order без исходного словаря

Октеты (в виде шестнадцатеричных символов) документа быстрого инфо-набора приведены в D.5.1. Подробные объяснения некоторых последовательностей октетов из D.5.1 приведены в D.5.2. Окончательные словари данного и предыдущего документов быстрого инфо-набора должны быть одинаковы, поскольку индексы словарных таблиц во внешнем словаре генерируются в том же порядке. Так как строки встроены в документ быстрого инфо-набора, то размер его будет больше. Затраты на включение строк — 635_{10} байт (размер данного документа быстрого инфо-набора минус размер предыдущего), что составляет примерно половину размера документа (для документов большего размера эта разница будет меньше, так как словарь имеет тенденцию к фиксированному размеру). В отличие от предыдущего документа быстрого инфо-набора данный документ можно рассматривать как самоописывающийся, потому что инфо-набор XML может быть создан без какой-либо внешней информации (внешнего словаря).

D.5.1 Октеты документа быстрого инфо-набора (в виде шестнадцатеричных символов)

В таблице D.8 приведены октеты документа быстрого инфо-набора для примера UBL Order, приведенного в D.3.

Примечание — В таблице D.8 подчеркнуты шестнадцатеричные символы, содержащие биты, которые соответствуют идентификации и указателю конца информационных элементов.

Таблица D.8 — Октеты документа быстрого инфо-набора (в виде шестнадцатеричных символов)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e001000000<u>78cf</u>027265733e75726e3a6f617369733a6e616d65733a74633a75
000020	626c3a636f64656c6973743a416<u>36</u>b6e6f776c656467656d656e74526573706f
000040	6e7365436f64653a313a30<u>cf</u>026362632f75726e3a6f617369733a6e616d6573
000060	3a74633a75626c3a436f6d6d6f6e426173696436f6d706f6e656e74733a313a
000080	30cf026361633375726e3a6f617369733a6e616d65733a74633a75626c3a436f
0000a0	6d6d6f6e4167677265676174656<u>36</u>f6d706f6e656e74733a313a30<u>cf</u>02637572
0000c0	2f75726e3a6f617369733a6e616d65733a74633a75626c3a636f64656c697374
0000e0	3a43757272656e6379436f64653a313a30<u>cf</u>027873692868747470703a2f2f7777
000100	772e77332e6f72672f323030312f584d4c53636865d612d696e7374616e6365
000120	cd1f75726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a31
000140	3a30<u>f03d</u>86044f72646572<u>7b</u>85850d736368656d614c6f636174696f6e083b75
000160	726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a313a3020
000180	2e2e2f2e2e2f7873642f6d61696e646f632f55424c2d4f72646572d312e302e
0001a0	787364<u>f03d</u>8607427579657273494482075330332d303334323537<u>f03f</u>828208
0001c0	497373756544617465820732030332d30322d3033<u>f03f</u>838309427579657250
0001e0	617274793<u>f</u>8383045061727479<u>3f</u>8383085061727479 4e616d653<u>f</u>8282034e61
000200	6d6582<u>0</u>e4a6572727920427569 6c6465722070 6c63<u>ff3f</u>838306416464726573
000220	733<u>f</u>828209537472656574e616d6582074d6172736204c616e65<u>f03f</u>828207
000240	436974794e616d6582044e6f7768657265<u>f03f</u>828209506f7374616c5a6f6e65
000260	<u>82</u>054e5231382034585<u>8f03f</u>82820f436f756e7472795375626564e7469747982
000280	044e6f72666f6c6<u>bff3f</u>838306436f6e7461637406820645766120427269636b
0002a0	<u>ffff3f</u>83830a53656c6c65725061727479040506821353706563696 16c697374

Окончание таблицы D.8

0002c0	2057696e646f777320706c63ff073f82820b4275696c64696e674e616d65820b
0002e0	536e6f7768696c6c20576f726b73f009820b4c6974746c6520536e6f72696e67
000300	f00a8204534d3220334e57f00b82075768657265736869726 5ffff3f83830744
000320	656c69766572793f82821852657175657374656444656c697665727944617465
000340	54696d658210323030332d30322d32345430303a30303a3030f03f83830e4465
000360	6c697665727941646472657373088 20a52697665727369646520526 42ef00e82
000380	17506c6f742031372c205768697465776174657220457374617465f009820657
0003a0	68657473746f6e65f00b82064d6964646c65736578fff03f8383084f72646572
0003c0	4c696e653f8383074c696e654974656d3f8383829041f07f8282075175616e74
0003e0	697479780f7175616e74697479556e6974436f646543756e6974f09032f03f83
000400	83034974656d3f83831853656c6c6572734974656d4964656e74696669636174
000420	696f6e3f838301494492023233365756f03f83831050687973696361 6c417474
000440	7269627574653f83830a41747472696275746549449201776f6f64f03f82820a
000460	4465736372697074696f6e9201736f6674ff191a820366696e697368f01b8203
000480	7072696d6564ff191a820566697474696e6773f01b9202736174696eff191a82
0004a0	04676c617a696e67f01b820373696e676c65ffffff1213149042f0550180f090
0004c0	33f016171892023334305457f0191a920168616e64f01b915248ff191aa3f01b
0004e0	920168617264ff191a820366696e697368f01b9202737461696eff191a820566
000500	697474696e6773f01b92026272617373ff191a8204676c617a696e67f01b8203
000520	646f75626c65ffffffff
00052a	

D.5.2 Объяснение кодирования

D.5.2.1 Кодирование информационных элементов document и Order element

Ниже приведено объяснение деталей исходного кодирования документа быстрого инфо-набора и корневого информационного элемента. В частности, объяснено кодирование информационного элемента **document**, последовательности информационных элементов **namespace**, информационных элементов **element** и **attribute**. В таблице D.9 приведен фрагмент документа быстрого инфо-набора для кодирования информационных элементов **document** и **Order element** из D.3.2. В таблице D.10 приведены детали данного кодирования. Фрагмент в XML 1.0 выглядит следующим образом:

```
<Order xmlns:res=«urn:oasis:names:tc:ubl:codelist:AcknowledgementResponseCode:1:0»
xmlns:cbc=«urn:oasis:names:tc:ubl:CommonBasicComponents:1:0»
xmlns:cac=«urn:oasis:names:tc:ubl:CommonAggregateComponents:1:0»
xmlns:cur=«urn:oasis:names:tc:ubl:codelist:CurrencyCode:1:0»
xmlns:xsi=«gost.e-ecolog.ru
xmlns=«urn:oasis:names:tc:ubl:Order:1:0»
xsi:schemaLocation=«urn:oasis:names:tc:ubl:Order:1:0  ../../xsd/maindoc/UBL-Order-
1.0.xsd»>
```

Таблица D.9 — Октеты фрагмента (в виде шестнадцатеричных символов)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e00100000078cf027265733e75726e3a6f617369733a6e616d65733a74633a75
000020	626c3a636f64656c6973743a41636b6e6f776c656467656d656e74526573706f
000040	6e7365436f64653a313a30cf026362632f75726e3a6f617369733a6e616d6573
000060	3a74633a75626c3a436f6d6d6f6e4261736963436f6d706f6e656e74733a313a
000080	30cf026361633375726e3a6f617369733a6e616d65733a74633a75626c3a436f
0000a0	6d6d6f6e41676772656761746543 6f6d706f6e656e74733a313a30cf02637572
0000c0	2f75726e3a6f617369733a6e616d65733a74633a75626c3a636f64656c697374
0000e0	3a43757272656e6379436f64653a313a30cf027873692868747470 3a2f2f7777
000100	772e77332e6f72672f323030312f584d4c536368656d612d696e7374616e6365
000120	cd1f75726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a31
000140	3a30f03d86044f726465727b85850d736368656d614c6f636174696f6e083b75
000160	726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a313a3020
000180	2e2e2f2e2e2f7873642f6d61696e646f632f55424c2d4f726465722d312e302e
0001a0	787364f0

Таблица D.10 — Детали кодирования

	Октет(ы)	Описание	Инфо-набор XML или XML
00 01	11100000 (e0) 00000000 (00)	Эти октеты присутствуют в начале каждого документа быстрого инфо-набора (см. 12.6)	Информационный элемент **document**
02 03	00000000 (00) 00000001 (01)	Эти октеты являются кодированием номера версии (см. 12.9)	
04	00000000 (00)	Эти октеты являются кодированием наличия исходного словаря и других компонентов типа **Document**. Октет на позиции 04_{16} со значением 00_{16} имеет значение '0' (забивка) в первом бите (см. 12.8). Биты со второго по восьмой содержат '0000000', обозначающие отсутствие всех опциональных компонентов типа **Document** (включая компонент **initial-vocabulary**, отсутствие которого обозначено третьим битом, см. C.2.3)	
05	01111000 (78)	Этот октет является исходным кодированием дочернего элемента информационного элемента **document**. Октет на позиции 05_{16} со значением 78_{16} имеет значение '0' (идентификация) для первого бита, обозначая, что у информационного элемента **document** есть дочерний элемент и им является информационный элемент **element** (см. C.2.11.2). Второй бит равен '1', обозначая, что информационный элемент **element** имеет атрибуты (см. C.3.3). Биты с третьего по шестой равны '1110' с последующими '00' (забивка) в битах с седьмого по восьмой, обозначая наличие пространства имен информационных элементов **attribute** (см. C.3.4.1)	Информационный элемент **element** со свойством **[namespace attribute]**

Продолжение таблицы D.10

	Октет(ы)	Описание	Инфо-набор XML или XML
06 07 08 .... 0a 0b 0c 4a	11001111 (cf) 00000010 (02) 01110010 (72) .... 01110010 (73) 00111110 (3e) 01110101 (75) 01110000 (30)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с литеральными свойствами **[prefix]** и **[normalized value]**. Октет на позиции 06_{16} со значением cf_{16} имеет значение '110011' (идентификация) в битах с первого по шестой, обозначая наличие пространства имен информационного элемента **attribute** (см. С.3.4.2). Седьмой бит равен '1', обозначая наличие свойства **[prefix]**. Восьмой бит равен '1', обозначая наличие свойства **[normalized value]**. Октет на позиции 07_{16} со значением 02_{16} имеет '0' в первом бите, обозначая, что закодирована литеральная строка символов для свойства **[prefix]** (см. С.13.3). Второй бит равен '0', обозначая, что длина закодированных символов UTF-8 больше или равна 1_{10} и меньше или равна 64_{10} и закодирована в битах с третьего по восьмой как целое без знака (см. С.22.3.1). Целое без знака равно 2_{10}, и длина равна 3_{10} (нижняя граница равна 1_{10}). Октеты 3_{10} с закодированными символами UTF-8 (свойства **[prefix]**), расположены с октета на позиции 08_{16} до октета на позиции $0a_{16}$. Строка «res» будет добавлена в таблицу PREFIX (с индексом 2_{10}). Октет на позиции $0b_{16}$ со значением $3e_{16}$ имеет '0' в первом бите, обозначая, что закодирована литеральная строка символов для свойства **[normalized value]** (см. С.13.3). Второй бит равен '0', обозначая, что длина закодированных символов UTF-8 больше или равна 1_{10} и меньше или равна 64_{10} и закодирована в битах с третьего по восьмой как целое без знака (см. С.22.3.1). Целое без знака равно 62_{10} и длина равна 63_{10} (нижняя граница равна 1_{10}). Октеты 63_{10}, с закодированными символами UTF-8 (свойства **[normalized value]**), расположены с октета на позиции $0c_{16}$ до октета на позиции $4a_{16}$. Строка «....ResponseCode:1:0» будет добавлена в таблицу NAMESPACE NAME (с индексом 2_{10})	**xmlns:res= «....ResponseCode:1:0»**
4b 4c 4d 4f 50 51 80	11001111 (cf) 00000010 (02) 01100011 (63) .... 01100011 (63) 00101111 (2f) 01110101 (75) 01110000 (30)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с литеральными свойствами **[prefix]** и **[normalized value]**. Октеты 3_{10} с закодированными символами UTF-8 (свойства **[prefix]**) расположены с октета на позиции $4c_{16}$ до октета на позиции $4f_{16}$. Строка «cbc» будет добавлена в таблицу PREFIX (с индексом 3_{10}). 48_{10} октетов закодированных символов UTF-8 (свойства **[normalized value]**) расположены с октета на позиции 51_{16} до октета на позиции 80_{16}. Строка «....sicComponents:1:0» будет добавлена в таблицу NAMESPACE NAME (с индексом 3_{10})	**xmlns:cbc= «....sicComponents:1:0»**
81 82 83 .... 85 86 87 .... ba	11001111 (cf) 00000010 (02) 01100011 (63) .... 01100011 (63) 00110011 (33) 01110101 (75) 01110000 (30)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с литеральными свойствами **[prefix]** и **[normalized value]**. Октеты 3_{10} с закодированными символами UTF-8 (свойства **[prefix]**) расположены с октета на позиции 83_{16} до октета на позиции 85_{16}. Строка «cac» будет добавлена в таблицу PREFIX (с индексом 4_{10}). Октеты 52_{10} с закодированными символами UTF-8 (свойства **[normalized value]**) расположены с октета на позиции 87_{16} до октета на позиции ba_{16}. Строка «....ateComponents:1:0» будет добавлена в таблицу NAMESPACE NAME (с индексом 4_{10})	**xmlns:cac= «....ateComponents:1:0»**

Продолжение таблицы D.10

	Октет(ы)	Описание	Инфо-набор XML или XML
bb bc bd .... bf c0 c1 .... f0	11001111 (cf) 00000010 (02) 01100011 (63) .... 01100011 (63) 00101111 (2f) 01110101 (75) 01110000 (30)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с литеральными свойствами **[prefix]** и **[normalized value]**. Октеты 3_{10} с закодированными символами UTF-8 (свойства **[prefix]**), расположены с октета на позиции bd_{16} до октета на позиции bf_{16}. Строка «cur» будет добавлена в таблицу PREFIX (с индексом 5_{10}). Октеты 48_{10} с закодированными символами UTF-8 (свойства **[normalized value]**), расположены с октета на позиции $c1_{16}$ до октета на позиции $f0_{16}$. Строка «….CurrencyCode:1:0» будет добавлена в таблицу NAMESPACE NAME (с индексом 5_{10})	**xmlns:cur= «….CurrencyCode:1:0»**
f1 f2 f3 .... f5 f6 f7 .... 11f	11001111 (cf) 00000010 (02) 01111000 (78) .... 01101001 (69) 00101000 (28) 01101000 (68) 01110111 (77)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с литеральными свойствами **[prefix]** и **[normalized value]**. Октеты 3_{10} с закодированными символами UTF-8 (свойства **[prefix]**) расположены с октета на позиции $f3_{16}$ до октета на позиции $f5_{16}$. Строка «xsi» будет добавлена в таблицу PREFIX (с индексом 6_{10}). Октеты 41_{10} с закодированными символами UTF-8 (свойства **[normalized value]**) расположены с октета на позиции $f7_{16}$ до октета на позиции $11f_{16}$. Строка «….Schema-instance» будет добавлена в таблицу NAMESPACE NAME (с индексом 6_{10})	**xmlns:xsi= «….Schema-instance»**
120 121 122 141	11001101 (cd) 00011111 (1f) 01110101 (75) 00110000 (30)	Эти октеты являются кодированием пространства имен информационного элемента **attribute** с индексированным свойством **[normalized value]**. Октеты 32_{10} с закодированными символами UTF-8 (свойства **[normalized value]**) расположены с октета на позиции 122_{16} до октета на позиции 141_{16}. Строка «….Order:1:0» будет добавлена в таблицу NAMESPACE NAME (с индексом 7_{10})	**xmlns=«…. Order:1:0»**
142	11110000 (f0)	Этот октет является кодированием указателя конца для последовательности пространств имен информационного элемента **attribute**. Октет на позиции 142_{16} со значением $f0_{16}$ имеет значение '1111' (указатель конца) в первых четырех битах (с первого по четвертый), и эти биты являются указателем конца последовательности. Четыре из шести значений '0' (забивка) содержатся в битах с пятого по восьмой (см. С.3.4.3)	
143 144 145 146 .... 14a	00111101 (3d) 10000110 (86) 00000100 (04) 01001111 (4f) 01110010 (72)	Эти октеты являются кодированием литерального квалифицированного имени информационного элемента **element**. Октет на позиции 143_{16} со значением $3d_{16}$ содержит два оставшихся из шести '0' (забивка) в первом и втором битах (см. С.3.4.3). Биты с третьего по шестой содержат значение '1111', обозначая, что квалифицированное имя является литеральным (см. С.18.3). Седьмой бит равен '0', обозначая, что у квалифицированного имени нет свойства **[prefix]**. Восьмой бит равен '1', обозначая наличие у квалифицированного имени свойства **[namespace name]**. Октет на позиции 144_{16} со значением 86_{16} имеет '1' в первом бите, обозначая, что свойство **[namespace name]** является не литеральным, а индексированным (см.С.13.4). Второй бит равен '0', обозначая, что индекс больше или равен 1_{10} и меньше или равен 64_{10} и закодирован в битах с третьего по восьмой как целое без знака (см . С.25.2). Целое без знака равно 6_{10}, и индекс равен 7_{10} (нижняя граница равна 1_{10}), что приводит к свойству **[namespace name]** «….Order:1.0» после перенаправления из таблицы NAMESPACE NAME.	**<Order ….**

Окончание таблицы D.10

	Октет(ы)	Описание	Инфо-набор XML или XML
		Октет на позиции 145_{16} со значением 04_{16} имеет '0' в первом бите, обозначая, что для свойства **[local name]** закодирована литеральная строка (см. С.13.3). Второй бит равен '0', обозначая, что длина закодированных символов UTF-8 больше или равна 1_{10} и меньше или равна 64_{10} и закодирована в битах с третьего по восьмой как целое без знака (см . С.22.3.1). Целое без знака равно 4_{10}, и длина равна 5_{10} (нижняя граница равна 1_{10}). Октеты 5_{10}, с закодированными символами UTF-8 (свойства **[local name]**) расположены с октета на позиции 146_{16} до октета на позиции $14a_{16}$. Строка «Order» будет добавлена в таблицу LOCAL NAME (с индексом 1_{10}). Квалифицированное имя без свойства **[prefix]**, со значением свойства **[namespace name]**, равным «....Order:1:0» (индекс 7_{10}), и свойства **[local name]**, равным «Order» (индекс 1_{10}), будет добавлено в таблицу ELEMENT NAME (с индексом 1_{10})	
14b 14c 14d 14e 14f 15c 15d 15e 15f	01111011 (7b) 10000101 (85) 10000101 (85) 00001101 (0d) 01110011 (73) 01101110 (6e) 00001000 (08) 00111011 (3b) 01110101 (75)	Эти октеты являются кодированием информационного элемента **attribute** с литеральным квалифицированным именем и свойством **[normalized value]**. Наличие информационных элементов **attribute** было обозначено в октете на позиции 05_{16} (второй бит равен '1'). Октет на позиции $14b_{16}$ со значением $7b_{16}$ имеет **[prefix]**, равным «xsi» (индекс 6_{10}), свойства **[namespace name]** — «....Schema-instance» (индекс 6_{10}) и свойства **[local name]** — «schemaLocation» (индекс 2_{10}), будет добавлено в таблицу ATTRIBUTE NAME (с индексом 1_{10}). Октет на позиции $15d_{16}$ со значением 08_{16} является исходным кодированием неидентифицирующей строки или индекса (см. С.14) для свойства **[normalized value]**. Первый бит равен '0', обозначая наличие литеральной строки символов (см. С.14.3). Второй бит равен '0', обозначая, что литеральную строку символов не следует добавлять в таблицу ATTRIBUTE VALUE. Третий и четвертый биты равны '0', обозначая, что форматом кодирования строки является UTF-8 (см. С.19.3.1). Пятый и шестой биты равны '1' и '0' соответственно, обозначая, что длина закодированных символов UTF-8 (свойство **[normalized value]**) больше или равна 9_{10} и меньше или равна 264_{10} и длина минус нижняя граница закодирована в восьми битах последующего октета как целое без знака (см. С.22.3.2). Биты с седьмого по восьмой равны '0' (забивка) (см. С.22.3.2). Октет на позиции $15e_{16}$ со значением $3b_{16}$ является кодированием целого без знака. Длина в октетах закодированных символов UTF-8 равна 68_{10} (нижняя граница равна 9_{10}). Октеты 68_{10}, с закодированными символами UTF-8 (свойства **[normalized value]**), расположены с октета на позиции $15f_{16}$ до октета на позиции $1a2_{16}$	**xsi:schemaLocation=«....»**
1a3	11110000 (f0)	Этот октет является кодированием указателя конца для последовательности информационных элементов **attribute**. Октет на позиции $1a3_{16}$ со значением $f0_{16}$ содержит '1111' в первых четырех битах (с первого по четвертый бит), что является указателем конца последовательности. Четыре '0' (забивка) присутствуют (с пятого по восьмой бит), так как информационный элемент **Order element** имеет дочерние элементы (см. D.3.2)	

D.5.2.2 Кодирование информационного элемента Address element информационного элемента BuyerParty element

Ниже приведено объяснение деталей кодирования информационного элемента **Address element** информационного элемента **BuyerParty element** документа быстрого инфо-набора. В частности, объяснено кодирование информационных элементов **element** и **character**. В таблице D.11 приведен фрагмент документа быстрого инфо-набора для кодирования информационного элемента **Address element**

информационного элемента **BuyerParty element** (D.3.2). В таблице D.12 приведено описание деталей кодирования. Фрагмент в XML 1.0 выглядит следующим образом:

```
<cac:Address>
        <cbc:StreetName>Marsh Lane</cbc:StreetName>
        <cbc:CityName>Nowhere</cbc:CityName>
        <cbc:PostalZone>NR18 4XX</cbc:PostalZone>
        <cbc:CountrySubentity>Norfolk</cbc:CountrySubentity>
</cac:Address>
```

Таблица D.11 — Октеты фрагмента (в виде шестнадцатеричных символов)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000200	3f838306416464726573
000220	733f828209537472656574e616d65820740617273820 4c616e65f03f828207
000240	436974794e616d6582044e6f7768657265f03f828209506f737461c5a6f6e65
000260	82054e523138203458 58f03f82820f436f756e74279537562656e7469747982
000280	044e6f72666f6c6bff

Таблица D.12 — Детали кодирования

	Октет(ы)	Описание	Инфо-набор XML или XML
216 217 218 219 21a 220	00111111 (3f) 10000011 (83) 10000011 (83) 00000110 (06) 01000001 (41) 01110011 (73)	Эти октеты являются кодированием информационного элемента **Address element**. Октет на позиции 216_{16} со значением $3f_{16}$ имеет '0' (идентификация) в первом бите, обозначая, что у информационного элемента **element** есть дочерний элемент (дочерний информационного элемента **Party element**), который является информационным элементом **element** (см. С.3.7.2). Второй бит равен '0', обозначая, что у информационного элемента **element** нет атрибутов (см. С.3.3). Биты с третьего по шестой равны '1111', обозначая, что квалифицированное имя является литеральным (см. С.18.3). Седьмой бит равен '1', обозначая, что у квалифицированного имени есть свойство **[prefix]**. Восьмой бит равен '1', обозначая, что у квалифицированного имени есть свойство **[namespace name]**. Октет на позиции 217_{16} со значением 83_{16} имеет первый бит '1', обозначая, что свойство **[prefix]** является не литеральным, а индексированным (см. С.13.4). Второй бит равен '0', обозначая, что индекс больше или равен 1_{10} и меньше или равен 64_{10} и закодирован в битах с третьего по восьмой как целое без знака (см. С.25.2). Целое без знака равно 3_{10}, и индекс равен 4_{10} (нижняя граница равна 1_{10}), что после перенаправления из таблицы PREFIX дает значение свойства **[prefix]**, равное «cac». Октет на позиции 218_{16} со значением 83_{16} имеет первый бит '1', обозначая, что свойство **[namespace name]** является не литеральным, а индексированным (см. С.13.4). Второй бит равен '0', обозначая, что индекс больше или равен 1_{10} и меньше или равен 64_{10} и закодирован в битах с третьего по восьмой как целое без знака (см. С.25.2). Целое без знака равно 3_{10}, и индекс равен 4_{10} (нижняя граница равна 1_{10}), что после перенаправления из таблицы NAMESPACE NAME дает значение свойства **[namespace name]**, равное «….ateComponents:1:0». Октет на позиции 219_{16} со значением 06_{16} имеет '0' в первом бите, обозначая, что для свойства **[local name]** закодирована литеральная строка (см. С.13.3). Второй бит равен '0', обозначая, что длина закодированных символов UTF-8 больше или равна 1_{10} и меньше или равна 64_{10} и закодирована в битах с третьего по восьмой как целое без знака (см. С.22.3.1). Целое без знака равно 6_{10}, и длина равна 7_{10} (нижняя граница равна 1_{10})	**<cac:Address>**

Продолжение таблицы D.12

	Октет(ы)	Описание	Инфо-набор XML или XML
		7_{10} октетов закодированных символов UTF-8 (свойства **[local name]**) расположены с октета на позиции $21a_{16}$ до октета на позиции 220_{16}. Строка «Address» будет добавлена в таблицу LOCAL NAME (с индексом 9_{10}). Квалифицированное имя со значением свойства **[prefix]**, равным «cac» (индекс 4_{10}), свойства **[namespace name]** — «.....ateComponents:1:0» (индекс 4_{10}) и свойства **[local name]** — «Order» (индекс 1_{10}), будет добавлено в таблицу ELEMENT NAME (с индексом 1_{10})	
221 222 223 224 225 .... 22e	00111111 (3f) 10000010 (82) 10000010 (82) 00001001 (09) 01000001 (53) 01100101 (65)	Эти октеты являются кодированием информационного элемента **StreetName element**. Свойство **[local name]** «StreetName» будет добавлено в таблицу LOCAL NAME (с индексом 10_{10}). Квалифицированное имя со значением свойства **[prefix]**, равным «cbc» (индекс 3_{10}), свойства **[namespace name]** — «.....BasicComponents:1:0» (индекс 3_{10}) и свойства **[local name]** — «StreetName» (индекс 10_{10}), будет добавлено в таблицу ELEMENT NAME (с индексом 9_{10})	<cbc:Street-Name>
22f 230 231 .... 23a	10000010 (82) 00000111 (07) 01001101 (4d) 01100101 (65)	Эти октеты являются кодированием информационных элементов **character** информационного элемента **StreetName element**. Октет на позиции $22f_{16}$ со значением 82_{16} имеет '10' (идентификация) в первых двух битах (с первого по второй биты), обозначая, что у информационного элемента **element** есть дочерний элемент (дочерний информационного элемента **StreetName element**), который является блоком информационных элементов **character** (см. С.3.7.5). Третий бит равен '0', обозначая наличие литеральной строки символов (см. С.15.3). Четвертый бит равен '0', обозначая, что литеральную строку символов не следует добавлять в таблицу CONTENT CHARACTER CHUNK. Пятый и шестой биты равны '0', обозначая, что форматом кодирования блока является UTF-8 (см. С.20.3.1). Седьмой и восьмой биты равны '1' и '0' соответственно, обозначая, что длина октетов закодированных символов UTF-8 (блока информационных элементов **character**) больше или равна 3_{10} и меньше или равна 258_{10} октетам и длина минус нижняя граница закодирована в восьми битах следующего октета как целое без знака (см. С.23.3.2). Октет на позиции 230_{16} со значением 07_{16} является кодированием целого без знака. Длина в октетах закодированных символов UTF-8 равна 10_{10} (нижняя граница равна 3_{10}). 10_{10} октетов закодированных символов UTF-8 расположены с октета на позиции 231_{16} до октета на позиции $23a_{16}$	Информационные элементы **character** «Marsh Lane»
23b	11110000 (f0)	Этот октет является указателем конца информационного элемента **StreetName element**	**</cbc:Street-Name>**
23c 23d 23e 23f 240 247	00111111 (3f) 10000010 (82) 10000010 (82) 00000111 (07) 01000011 (43) 01100101 (65)	Эти октеты являются кодированием информационного элемента **CityName element**. Свойство **[local name]** «CityName» будет добавлено в таблицу LOCAL NAME (с индексом 11_{10}). Квалифицированное имя со значением свойства **[prefix]**, равным «cbc» (индекс 3_{10}), свойства **[namespace name]** — «.....BasicComponents:1:0» (индекс 3_{10}) и свойства **[local name]** — «CityName» (индекс 11_{10}), будет добавлено в таблицу ELEMENT NAME (с индексом 10_{10})	**<cbc:City-Name>**
248 249 24a 250	10000010 (82) 00000100 (04) 01001110 (4e) 01100101 (65)	Эти октеты являются кодированием информационных элементов **character** из информационного элемента **CityName element**. 7_{10} октетов закодированных символов UTF-8 начинаются с октета на позиции $24a_{16}$ до октета на позиции 250_{16}	Информационные элементы **character** «Nowhere»
251	11110000 (f0)	Этот октет является указателем конца информационного элемента **CityName element**	**</cbc:City-Name>**

Окончание таблицы D.12

	Октет(ы)	Описание	Инфо-набор XML или XML
252 253 254 255 256 .... 25f	00111111 (3f) 10000010 (82) 10000010 (82) 00001001 (09) 01000011 (50) 01100101 (65)	Эти октеты являются кодированием информационного элемента **PostalZone element**. Свойство **[local name]** «PostalZone» будет добавлено в таблицу LOCAL NAME (с индексом 12_{10}). Квалифицированное имя со значением свойства **[prefix]**, равным «cbc» (индекс 3_{10}), свойства **[namespace name]** – «.....BasicComponents:1:0» (индекс 3_{10}) и свойства **[local name]** – «PostalZone» (индекс 12_{10}), будет добавлено в таблицу ELEMENT NAME (с индексом 11_{10})	**<cbc:Postal-Zone>**
260 261 262 269	10000010 (82) 00000101 (05) 01001110 (4e) 01011000 (58)	Эти октеты являются кодированием информационных элементов **character** из информационного элемента **PostalZone element**. 8_{10} октетов закодированных символов UTF-8 начинаются с октета на позиции 262_{16} до октета на позиции 269_{16}	Информационные элементы **character** «NR18 4XX»
26a	11110000 (f0)	Этот октет является указателем конца информационного элемента **PostalZone element**	**</cbc:Postal-Zone>**
26b 26c 26d	00111111 (3f) 10000010 (82) 10000010 (82)	Эти октеты являются кодированием информационного элемента **CountrySubentity element**	**<cbc:Country-Subentity>**
26e 26f 27e	00001111 (0f) 01000011 (43) 01111001 (79)	Свойство **[local name]** «CountrySubentity» будет добавлено в таблицу LOCAL NAME (с индексом 13_{10}). Квалифицированное имя со значением свойства **[prefix]**, равным «cbc» (индекс 3_{10}), свойства **[namespace name]** — «.....BasicComponents:1:0» (индекс 3_{10}) и свойства **[local name]** — «CountrySubentity» (индекс 13_{10}), будет добавлено в таблицу ELEMENT NAME (с индексом 12_{10})	
27f 280 281 287	10000010 (82) 00000100 (04) 01001110 (4e) 01101011 (6b)	Эти октеты являются кодированием информационных элементов **character** из информационного элемента **CountrySubentity element**. 7_{10} октетов закодированных символов UTF-8 начинаются с октета на позиции 281_{16} до октета на позиции 287_{16}	Информационные элементы **character** «Norfolk»
288	11111111 (ff)	Этот октет является указателем конца информационных элементов **CountrySubentity element** и **Address element**. Октет на позиции 288_{16} со значением ff_{16} содержит '1111' в первых четырех битах (с первого по четвертый бит), что является указателем конца информационного элемента **CountrySubentity element**. (см. С.3.8). Последние четыре бита (биты с пятого по восьмой) содержат '1111' и являются указателем конца информационного элемента **Address element** (см. С.3.8)	**</cbc:CountrySubentity></cac:Address>**

D.5.2.3 Кодирование информационного элемента BuyersID element первого информационного элемента LineItem element

Ниже приведено объяснение деталей кодирования информационного элемента **BuyersID element** первого информационного элемента **LineItem element** документа быстрого инфо-набора. В частности, объяснено кодирование информационного элемента **element**, свойство **[local name]** которого было проиндексировано ранее. В таблице D.13 приведен фрагмент документа быстрого инфо-набора для кодирования информационного элемента **BuyersID element** первого информационного элемента **LineItem element** (D.3.2). В таблице D.14 приведено описание деталей кодирования. Фрагмент в XML 1.0 выглядит следующим образом:

```
<cac:LineItem>
        <cac:BuyersID>A</cac:BuyersID>
```

Таблица D.13 — Октеты фрагмента (в виде шестнадцатеричных символов)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
0003c0	3f8383074c696e654974656d3f83838290 41f0

Таблица D.14 — Детали кодирования

	Октет(ы)	Описание	Инфо-набор XML или XML
3c4 3c5 3c6 3c7 3c8	00111111 (3f) 10000011 (83) 10000011 (83) 00001111 (07) 01001010 (4c)	Эти октеты являются кодированием информационного элемента **LineItem element**. Свойство **[local name]** «LineItem» будет добавлено в таблицу LOCAL NAME (с индексом 21_{10})	**<cac:LineItem>**
.... 3cf	.... 01101101 (6d)	Квалифицированное имя со значением свойства **[prefix]**, равным «cac» (индекс 4_{10}), свойства **[namespace name]** — «.....ateComponents:1:0» (индекс 4_{10}) и свойства **[local name]** — «LineItem» (индекс 10_{10}), будет добавлено в таблицу ELEMENT NAME (с индексом 20_{10})	
3d0 3d1 3d2 3d3	00111111 (3f) 10000011 (83) 10000011 (83) 10000010 (82)	Эти октеты являются кодированием информационного элемента **BuyersID element**. Свойства **[prefix]**, **[namespace name]** и **[local name]** уже были проиндексированы, так как все связанные с ними строки уже встречались ранее до этого информационного элемента. Свойство **[local name]** было проиндексировано при обработке первого дочернего элемента информационного элемента **Order element**, а именно информационного элемента **BuyersID element** со значением свойства **[namespace name]** «....Order:1:0». Квалифицированное имя со значением свойства **[prefix]**, равным «cac» (индекс 4_{10}), свойства **[namespace name]** — «.....ateComponents:1:0» (индекс 4_{10}) и свойства **[local name]** — «BuyersID» (индекс 3_{10}), будет добавлено в таблицу ELEMENT NAME (с индексом 21_{10})	**<cac:BuyersID>**
3d4 3d5	10010000 (90) 01000001 (41)	Эти октеты являются кодированием информационных элементов **character** информационного элемента **BuyersID element**. Октет на позиции $3d4_{16}$ со значением 90_{16} имеет '10' (идентификация) в первых двух битах (с первого по второй биты), обозначая, что у информационного элемента **element** есть дочерний элемент (дочерний информационного элемента **BuyersID element**), который является блоком информационных элементов **character** (см. С.3.7.5). Третий бит равен '0', обозначая наличие литеральной строки символов (см. С.15.3). Четвертый бит равен '1', обозначая, что литеральную строку символов следует добавить в таблицу CONTENT CHARACTER CHUNK (в данном примере строки длиной менее 6_{10} символов добавляют в таблицу CONTENT CHARACTER CHUNK или ATTRIBUTE VALUE). Пятый и шестой биты равны '0', обозначая, что форматом кодирования блока является UTF-8 (см. С.20.3.1). Седьмой бит равен '0', обозначая, что длина октетов закодированных символов UTF-8 (блока информационных элементов **character**) больше или равна 1_{10} и меньше или равна 2_{10} и длина минус нижняя граница закодирована в восьмом бите как целое без знака (см. С.24.3.1). Целое без знака равно 0_{10}, и длина равна 1_{10}. Октет закодированных символов UTF-8 находится в октете на позиции 41_{16}	Информационный элемент **character** «A»
3d6	11110000 (f0)	Этот октет является указателем конца информационного элемента **BuyersID element**	**</ cac:BuyersID>**

Приложение Е
(справочное)

Присвоение значений идентификаторов объектов

В настоящем стандарте присвоены следующие идентификаторы объектов и дескрипторы объектов:

```
{ joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoset(0)
modules(0) fast-infoset(0)}
«Fast Infoset ASN.1 Module»
{joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoset(0)
modules(0) fast-infoset-edm(1)}
«Fast Infoset Encoding Definition Module»
{joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoset(0)
modules(0)fast-infoset-elm(2)}
«Fast Infoset Encoding Link Module»
{joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoset(0)
encodings(1) optional-xml-declaration(0)}
-- Определен как finf-doc-opt-decl в A.1
«A fast infoset document containing an XML declaration»
{joint-iso-itu-t(2) asn1(1) generic-applications(10) fast-infoset(0)
encodings(1) no-xml-declaration(1)}
-- Определен как finf-doc-no-decl в A.1
«A fast infoset document without an XML declaration»
```

Приложение ДА
(справочное)

Сведения о соответствии межгосударственных стандартов ссылочным международным стандартам (международным документам)

Таблица ДА.1

Обозначение и наименование международного стандарта	Степень соответствия	Обозначение и наименование межгосударственного стандарта
ISO/IEC 9834-8:2005 Информационные технологии. Взаимосвязь открытых систем. Процедуры для работы регистрационных органов в системе OSI. Часть 8. Создание и регистрация универсально уникальных идентификаторов и их использование в качестве компонентов идентификаторов объектов ASN.1	—	*
ISO/IEC 8824-1:2002 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 1. Спецификация базовой нотации	—	*
ISO/IEC 8824-2:2002 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 2. Спецификация информационных объектов	—	*
ISO/IEC 8824-3:2002 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 3. Спецификация ограничений	—	*
ISO/IEC 8824-4:2002 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 4. Спецификация для параметризации ASN.1	—	*
ISO/IEC 8825-1:2002 Информационные технологии. Правила кодирования ASN.1. Часть 1. Спецификация основных (BER), канонических (CER) и различительных правил кодирования (DER)	—	*
ISO/IEC 8825-2:2002 Информационные технологии. Правила кодирования ASN.1. Часть 2. Спецификация правил уплотненного кодирования (PER)	—	*
ISO/IEC 8825-3:2002 Информационные технологии. Правила кодирования ASN.1. Часть 3. Спецификация нотации контроля кодирования (ECN)	—	*
ISO/IEC 8825-4:2002 Информационные технологии. Правила кодирования ASN.1. Часть 4. Правила кодирования XML (XER)	—	*
ISO 8601:2004 Элементы данных и форматы для обмена информацией. Обмен информацией. Представление дат и времени	—	*
ISO/IEC 10646:2003 Информационные технологии. Универсальный многооктетный набор кодированных символов (UCS)	—	*

* Соответствующий межгосударственный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Библиография

[1] OASIS Universal Business Language (UBL) 1.0 (Универсальный бизнес-язык UBL)

[2] IETF RFC 1952 (1996), GZIP file format specification version 4.3 (Формат файлов GZIP, спецификация версии 4.3)

УДК 681.3:691.39:006.354 МКС 35.100.60 IDT

Ключевые слова: обработка данных, информационный обмен, сетевое взаимодействие, взаимосвязь открытых систем, ASN.1, кодирование, быстрый инфо-набор

Редактор *Н.Н. Кузьмина*
Технический редактор *В.Н. Прусакова*
Корректор *Е.Р. Ароян*
Компьютерная верстка *И.В. Белюсенко*

Сдано в набор 09.11.2015. Подписано в печать 15.12.2015. Формат 60 × 84 $^{1}/_{8}$. Гарнитура Ариал.
Усл. печ. л. 12,09. Уч.-изд. л. 11,00. Тираж 32 экз. Зак. 18.

Набрано в ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.
www. jurisizdat. ru gost.e-ecolog.ru

Издано и отпечатано во
ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4
gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru