Международный союз электросвязи

МСЭ-Т

СЕКТОР СТАНДАРТИЗАЦИИ ЭЛЕКТРОСВЯЗИ
МСЭ

X.891

(05/2005)

СЕРИЯ X: СЕТИ ПЕРЕДАЧИ ДАННЫХ, ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ И БЕЗОПАСНОСТЬ

Приложения ВОС: Общие приложения ASN.1

Информационные технологии – Общие приложения ASN.1: Быстрый информационный набор (Fast Infoset)

Рекомендация МСЭ-Т X.891

ITU-T

Международный союз электросвязи

РЕКОМЕНДАЦИИ МСЭ-Т СЕРИИ X

СЕТИ ПЕРЕДАЧИ ДАННЫХ, ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ И БЕЗОПАСНОСТЬ

СЕТИ ПЕРЕДАЧИ ДАННЫХ ОБЩЕГО ПОЛЬЗОВАНИЯ	X.1–X.199
Службы и услуги	X.1–X.19
Интерфейсы	X.20–X.49
Передача, сигнализация и коммутация	X.50–X.89
Сетевые аспекты	X.90–X.149
Техническое обслуживание	X.150–X.179
Административные предписания	X.180–X.199
ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ	X.200–X.299
Модель и обозначение	X.200–X.209
Определения служб	X.210–X.219
Спецификации протоколов с установлением соединений	X.220–X.229
Спецификации протоколов без установления соединений	X.230–X.239
Проформы PICS	X.240–X.259
Идентификация протоколов	X.260–X.269
Протоколы обеспечения безопасности	X.270–X.279
Управляемые объекты уровня	X.280–X.289
Испытание на соответствие	X.290–X.299
ВЗАИМОДЕЙСТВИЕ МЕЖДУ СЕТЯМИ	X.300–X.379
Общие положения	X.300–X.349
Спутниковые системы передачи данных	X.350–X.369
Сети, основанные на протоколе Интернет	X.370–X.379
СИСТЕМЫ ОБРАБОТКИ СООБЩЕНИЙ	X.400–X.499
СПРАВОЧНИК	X.500–X.599
ОРГАНИЗАЦИЯ СЕТИ ВОС И СИСТЕМНЫЕ АСПЕКТЫ	X.600–X.699
Организация сети	X.600–X.629
Эффективность	X.630–X.639
Качество обслуживания	X.640–X.649
Наименование, адресация и регистрация	X.650–X.679
Абстрактно-синтаксическая нотация 1 (ASN.1)	X.680–X.699
УПРАВЛЕНИЕ В ВОС	X.700–X.799
Структура и архитектура управления системами	X.700–X.709
Служба и протокол связи для общего управления	X.710–X.719
Структура управляющей информации	X.720–X.729
Функции общего управления и функции ODMA	X.730–X.799
БЕЗОПАСНОСТЬ	X.800–X.849
ПРИЛОЖЕНИЯ ВОС	X.850–X.899
Фиксация, параллельность и восстановление	X.850–X.859
Обработка транзакций	X.860–X.879
Удаленные операции	X.880–X.889
Общие приложения ASN.1	**X.890–X.899**
ОТКРЫТАЯ РАСПРЕДЕЛЕННАЯ ОБРАБОТКА	X.900–X.999
БЕЗОПАСНОСТЬ ЭЛЕКТРОСВЯЗИ	X.1000–

Для получения более подробной информации просьба обращаться к перечню Рекомендаций МСЭ-Т.

МЕЖДУНАРОДНЫЙ СТАНДАРТ ИСО/МЭК 24824-1
РЕКОМЕНДАЦИЯ МСЭ-Т X.891

Информационные технологии – Общие приложения ASN.1: Быстрый информационный набор

Резюме

В настоящей Рекомендации | Международном стандарте определяется представление экземпляра Информационного набора XML W3C с использованием двоичных кодирований. Данные двоичные кодирования определяются при помощи нотации ASN.1 и Нотации контроля кодирования (ECN) ASN.1.

Технология, определенная в настоящей Рекомендации | Международном стандарте, называется Быстрый информационный набор (Fast Infoset). Она предоставляет альтернативный по отношению к XML W3C синтаксис для представления экземпляров Информационного набора XML W3C. Данное представление в общем случае обеспечивает меньшие размеры кодирований и более высокую скорость обработки по сравнению с представлением XML W3C.

В настоящей Рекомендации | Международном стандарте определяется использование ряда методик, позволяющих минимизировать размер кодирований (называемых документами Быстрого информационного набора) и максимизировать скорость создания и обработки документов Быстрого информационного набора. Данные методики включают использование динамических таблиц (как для строк символов, так и для уточненных имен), исходных словарей и внешних словарей.

В настоящей Рекомендации | Международном стандарте также определяется тип среды Многоцелевых расширений почтовой службы в Internet (MIME), идентифицирующий документ Быстрого информационного набора.

Источник

Рекомендация МСЭ-Т X.891 утверждена 14 мая 2005 года 17-й Исследовательской комиссией МСЭ-Т (2005–2008 гг.) в соответствии с процедурой, изложенной в Рекомендации МСЭ-Т A.8. Она содержит исправления, внесенные Техническим списком исправлений 1, утвержденным 13 июня 2006 года 17-й Исследовательской комиссией МСЭ-Т (2005–2008 гг.) в соответствии с процедурой, изложенной в Рекомендации МСЭ-Т A.8. Идентичный текст опубликован также в виде ИСО/МЭК 24824-1.

ПРЕДИСЛОВИЕ

Международный союз электросвязи (МСЭ) является специализированным учреждением Организации Объединенных Наций в области электросвязи. Сектор стандартизации электросвязи МСЭ (МСЭ-Т) – постоянный орган МСЭ. МСЭ-Т отвечает за изучение технических, эксплуатационных и тарифных вопросов и за выпуск Рекомендаций по ним с целью стандартизации электросвязи на всемирной основе.

На Всемирной ассамблее по стандартизации электросвязи (ВАСЭ), которая проводится каждые четыре года, определяются темы для изучения Исследовательскими комиссиями МСЭ-Т, которые, в свою очередь, вырабатывают Рекомендации по этим темам.

Утверждение Рекомендаций МСЭ-Т осуществляется в соответствии с процедурой, изложенной в Резолюции 1 ВАСЭ.

В некоторых областях информационных технологий, которые входят в компетенцию МСЭ-Т, необходимые стандарты разрабатываются на основе сотрудничества с ИСО и МЭК.

ПРИМЕЧАНИЕ

В настоящей Рекомендации термин "администрация" используется для краткости и обозначает как администрацию электросвязи, так и признанную эксплуатационную организацию.

Соблюдение положений данной Рекомендации носит добровольный характер. Однако в Рекомендации могут содержаться определенные обязательные положения (например, для обеспечения возможности взаимодействия или применимости), и соблюдение положений данной Рекомендации достигается в случае выполнения всех этих обязательных положений. Для выражения необходимости выполнения требований используется синтаксис долженствования и соответствующие слова (такие, как "должен" и т. п.), а также их отрицательные эквиваленты. Использование этих слов не предполагает, что соблюдение положений данной Рекомендации является обязательным для какой-либо из сторон.

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

МСЭ обращает внимание на вероятность того, что практическое применение или реализация этой Рекомендации может включать использование заявленного права интеллектуальной собственности. МСЭ не занимает какую бы то ни было позицию относительно подтверждения, обоснованности или применимости заявленных прав интеллектуальной собственности, независимо от того, отстаиваются ли они членами МСЭ или другими сторонами вне процесса подготовки Рекомендации.

На момент утверждения настоящей Рекомендации МСЭ получил извещение об интеллектуальной собственности, защищенной патентами, которые могут потребоваться для выполнения этой Рекомендации. Однако те, кто будет применять Рекомендацию, должны иметь в виду, что это может не отражать самую последнюю информацию, и поэтому им настоятельно рекомендуется обращаться к патентной базе данных БСЭ по адресу: http://www.itu.int/ITU-T/ipr/.

Содержание

Стр.

Введение

В настоящей Рекомендации | Международном стандарте определяется представление экземпляра Информационного набора XML W3C с использованием двоичных кодирований (определенных при помощи нотации ASN.1 и Нотации контроля кодирования (ECN) ASN.1). Кодирования, определенные в этом разделе настоящей Рекомендации | Международного стандарта, идентифицируются номером версии 1 (см. п. 12.9).

Технология, определенная в настоящей Рекомендации | Международном стандарте, называется Быстрый информационный набор (Fast Infoset). Она предоставляет альтернативный, по отношению к XML W3C, синтаксис для представления экземпляров Информационного набора XML W3C. Данное представление в общем случае обеспечивает меньшие размеры кодирования и более высокую скорость обработки по сравнению с представлением XML W3C.

Определенное в настоящей Рекомендации | Международном стандарте представление экземпляра Информационного набора XML W3C называется документом Быстрого информационного набора. Каждый документ Быстрого информационного набора представляет собой кодирование абстрактного значения типа данных ASN.1 (тип данных **`Document`** см. п. 7.2) представляющий экземпляр Информационного набора XML W3C.

В настоящей Рекомендации | Международном стандарте определяется использование ряда методик, позволяющих минимизировать размер документов Быстрого информационного набора, а также максимизировать скорость создания и обработки таких документов.

Данные методики основаны на использовании словарных таблиц, позволяющих использовать значения типа "малое целое" (индексы словарных таблиц) вместо строк символов, формирующих, например, имена элементов или атрибутов в сериализации XML 1.0 экземпляра Информационного набора XML W3C.

Существует целый ряд словарных таблиц (см. п. 8), самые основные из которых (таблицы 8-символьных строк) устанавливают соответствие между строками символов и значениями типа "малое целое". Однако существуют также и словарные таблицы (таблицы имен элементов и таблицы имен атрибутов), позволяющие осуществлять преобразования более высокого уровня, где индекс словарной таблицы соответствует набору из трех индексов словарной таблицы, идентифицирующих префикс, имя пространства имен и локальное имя.

Другая важная методика состоит в использовании словарных таблиц с ограниченным алфавитом. Такие таблицы содержат записи, перечисляющие подмножество символов ИСО/МЭК 10646. Если требуется закодировать строку символов, для которой есть запись в данной таблице, то она может быть закодирована при помощи идентификации того, что данная словарная таблица используется, путем указания индекса словарной таблицы с последующим кодированием каждого символа минимальным числом битов, необходимым для данного конкретного подмножества символов ИСО/МЭК 10646. Существует ряд встроенных ограниченных алфавитов, всегда формирующих несколько первых записей в таблице, которые охватывают такие часто встречающиеся строки как дата, время и числовые значения.

Следующим важным в плане оптимизации моментов является использование словарной таблицы алгоритма кодирования. Данная таблица указывает специализированные кодировки, которые могут быть применены к часто встречающимся строкам, также с рядом встроенных алгоритмов. Например, если имеется строка, выглядящая как десятичное представление целого числа в диапазоне от -32768 до 32767, то эта строка может быть закодирована при помощи идентификации того, что данная словарная таблица используется, путем указания индекса словарной таблицы с последующим кодированием целого числа как двухоктетного целого числа со знаком. Числа с плавающей точкой и массивы таких чисел поддерживаются аналогичным образом.

Для того чтобы обеспечить быструю обработку без необходимости жертвовать компактностью, многие компоненты документа Быстрого информационного набора (такие, как строки символов и компоненты, представляющие единицы информации Информационного набора XML) выравниваются по октетам, в то время как другие компоненты (такие, как длины и индексы словарных таблиц) не обязательно выравниваются по октетам, но всегда оканчиваются на последнем бите октета. Для предоставления формальной спецификации этих оптимизированных кодирований используется Нотация контроля кодирования ASN.1 (определена в Рек. МСЭ-Т X.692 | ИСО/МЭК 8825-3) (см. п. A.2), однако использование инструментов ECN для реализации не является необходимым и предоставляется полное описание кодирования (см. Приложение C).

Словарные таблицы для конкретного документа Быстрого информационного набора могут быть инициализированы с помощью информации в головной части документа и обычно добавляются динамически, обеспечивая гибкость кодировщику. Исходные словарные таблицы могут быть предоставлены посредством ссылки на окончательные словарные таблицы какого-либо другого идентифицированного документа Быстрого информационного набора (или другим способом). Данная словарная ссылка может быть впоследствии дополнена дальнейшими добавлениями в таблицу для того, чтобы предоставить исходные таблицы для данного документа. Дальнейшие динамические добавления в таблицы обычно осуществляются в процессе создания или обработки документа.

Наконец, предоставляется механизм включения создателем документа Быстрого информационного набора дополнительных данных (называемых дополнительными данными обработки), связанных с дополнительной обработкой документа Быстрого информационного набора наряду с URI (Унифицированным идентификатором ресурса), идентифицирующим полную спецификацию формы и значения этих дополнительных данных обработки. Необязательные дополнительные данные обработки игнорируются всеми последующими обработчиками документа Быстрого информационного набора в случае, если URI не известен или если указываемая им обработка не поддерживается или не требуется.

ПРИМЕЧАНИЕ. – Примером таких дополнительных данных обработки могут являться данные, предоставляющие индексы, позволяющие осуществлять немедленный доступ к частям документа Быстрого информационного набора так, что нет необходимости в обработке всего документа целиком в случае, если интерес представляют только части документа Быстрого информационного набора, соответствующие определенному тэгу XML.

В Приложении A, являющемся неотъемлемой частью настоящей Рекомендации | Международного стандарта, содержатся модуль ASN.1 (см. Рек. МСЭ-Т X.680 | ИСО/МЭК 8824-1) и два модуля ECN (EDM и ELM – см. Рек. МСЭ-Т X.692 | ИСО/МЭК 8825-3), которые вместе определяют абстрактное содержимое кодирования битового уровня значения типа **`Document`**, который передает значение экземпляра Информационного набора XML W3C.

В Приложении B, являющемся неотъемлемой частью настоящей Рекомендации | Международного стандарта, содержится спецификация типа среды Многоцелевых расширений почтовой службы в Internet (MIME), идентифицирующего документ Быстрого информационного набора.

В Приложении C, не являющемся неотъемлемой частью настоящей Рекомендации | Международного стандарта, приводится полное описание кодирований, формально определенных в п. 12 и п. A.2.

В Приложении D, не являющемся неотъемлемой частью настоящей Рекомендации | Международного стандарта, приводятся примеры документов Быстрого информационного набора, созданных из некоторых документов XML. В Приложении D также приводится размер представлений этих документов в форме XML и в форме Быстрого информационного набора.

МЕЖДУНАРОДНЫЙ СТАНДАРТ
РЕКОМЕНДАЦИЯ МСЭ-Т

Информационные технологии – Общие приложения ASN.1: Быстрый информационный набор

1 Область применения

В настоящей Рекомендации | Международном стандарте определяется тип ASN.1 (см.Рек. МСЭ-Т X.680 | ИСО/МЭК 8824-1) абстрактные значения которого представляют экземпляры Информационного набора XML W3C. Здесь также определяются двоичные кодирования для данных значений с использованием Нотации контроля кодирования ASN.1 (см. see Рек. МСЭ-Т X.692 | ИСО/МЭК 8825-3).

ПРИМЕЧАНИЕ. – Данные кодирования называются документами Быстрого информационного набора.

В настоящей Рекомендации | Международном стандарте также определяются методики, позволяющие:

- минимизировать размер документов Быстрого информационного набора;
- максимизировать скорость создания и обработки документов Быстрого информационного набора;
- определять (создателю документа Быстрого информационного набора) дополнительные данные обработки.

Первые две методики предусматривают использование концептуальных словарных таблиц. Набор словарных таблиц и сущность их записей полностью определены в настоящей Рекомендации | Международном стандарте, однако их представление в памяти компьютера выходит за рамки настоящей Рекомендации | Международного стандарта. Описание обеспечения передачи или хранения, а также формальной нотации для визуального отображения или определения словарных таблиц, которые следует использовать в качестве внешних словарей, также находятся за рамками настоящей Рекомендации | Международного стандарта.

Третья методика предусматривает предоставление дополнительных данных обработки и URI, который идентифицирует форму и значение этих данных. Спецификация конкретных форм дополнительных данных обработки и их использования выходит за рамки настоящей Рекомендации | Международного стандарта.

URI могут использоваться для идентификации окончательных словарей, которые могут быть использованы либо как часть какого-то исходного словаря либо составлять его целиком, однако назначение конкретных URI конкретным окончательным словарям выходит за рамки настоящей Рекомендации | Международного стандарта.

В настоящей Рекомендации | Международном стандарте определяются встроенные ограниченные алфавиты, добавление к словарным таблицам добавочных ограниченных алфавитов путем нумерации, а также использование этих словарных таблиц для эффективного кодирования строк символов.

В настоящей Рекомендации | Международном стандарте определяются встроенные алгоритмы кодирования для оптимального кодирования определенных строк символов и добавление в словарные таблицы добавочных алгоритмов кодирования, идентифицированных URI, однако определение этих добавочных алгоритмов и связанных с ними URI выходит за рамки настоящей Рекомендации | Международного стандарта.

Дополнительно, в настоящей Рекомендации | Международном стандарте определяется тип среды Многоцелевых расширений почтовой службы в Internet (MIME), идентифицирующий документ Быстрого информационного набора.

2 Нормативная справочная литература

Указанные ниже Рекомендации, Международные стандарты и прочие источники содержат положения, которые путем ссылки на них в данном тексте составляют положения настоящей Рекомендации | Международного стандарта. На момент публикации указанные издания были действующими. Все Рекомендации, Международные стандарты и прочие источники могут подвергаться пересмотру; поэтому сторонам соглашений, основанных на данной Рекомендации | Международном стандарте, предлагается изучить возможность применения последнего издания Рекомендаций, Международных стандартов и прочих источников, перечисленных ниже. Бюро стандартизации электросвязи МСЭ ведет список действующих в настоящее время Рекомендаций МСЭ-Т. Члены МЭК и ИСО ведут регистры действующих в настоящее время Международных стандартов. Рабочая группа по стандартам для сети Internet (IETF) ведет список Запросов на пояснение (RFC), включающий, также, и те устаревшие RFC, которые были заменены более новыми. Консорциум World-Wide Web (W3C) ведет список ведет список действующих в настоящее время Рекомендаций W3C. Ссылка на документ в рамках этой Рекомендации | Международного стандарта не дает ему, как отдельному документу, статуса Рекомендации | Международного стандарта.

2.1 Идентичные Рекомендации | Международные стандарты

– ITU-T Recommendation X.667 (2004) | ISO/IEC 9834-8:2005, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components.*

– ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

– ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.* †

– ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.* †

– ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.* †

– ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER).* †

– ITU-T Recommendation X.691 (2002) | ISO/IEC 8825-2:2002, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER).* †

– ITU-T Recommendation X.692 (2002) | ISO/IEC 8825-3:2002, *Information technology – ASN.1 encoding rules: Specification of Encoding Control Notation (ECN).*

– ITU-T Recommendation X.693 (2001) | ISO/IEC 8825-4:2002, *Information technology – ASN.1 encoding rules: XML Encoding Rules (XER).* †

ПРИМЕЧАНИЕ. – Выше приведен полный набор Рекомендаций | Международных стандартов ASN.1, поскольку все они могут быть применимы в конкретных случаях использования настоящей Рекомендации | Международного стандарта. В случае, если в тексте настоящей Рекомендации | Международного стандарта не содержится прямых ссылок на тот или иной документ, в приведенном выше списке к этому документу добавлен символ †.

2.2 Дополнительная справочная литература

– ISO 8601:2004, *Data elements and interchange formats – Information interchange – Representation of dates and times.*

– ISO/IEC 10646:2003, *Information technology – Universal Multiple-Octet Coded Character Set (UCS).*

– *The Unicode Standard, Version 4.0,* The Unicode Consortium *(Reading, MA, Addison-Wesley).*

ПРИМЕЧАНИЕ 1. –Графические символы (и их кодирования), определенные стандартом Unicode идентичны тем, которые определяются ИСО/МЭК 10646-1, однако стандарт Unicode включен в качестве справочного источника, поскольку он также определяет имена управляющих символов и определяет аббревиатуру 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), 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), http://www.w3.org/TR/2000/REC-xml11-20040204/.*

ПРИМЕЧАНИЕ 2. – Включены ссылки как на XML 1.0 W3C, так и на XML 1.1 W3C, поскольку ни один из них не является подмножеством другого. Данные ссылки используются только в п. 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), 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://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), http://www.w3.org/TR/2004/REC-xm-l-names11-20040204/.*

ПРИМЕЧАНИЕ 3. – Включены ссылки как на Пространства имен XML 1.0 W3C, так и на Пространства имен XML 1.1 W3C, поскольку ни одно из них не является подмножеством другого. Данные ссылки используются только в п. 3.4.10..

– IETF RFC 2045 (1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.*

– IETF RFC 2396 (1998), Uniform Resource Identifiers (URI): Generic Syntax.

– IEEE 754-1985, IEEE Standard for Binary Floating-Point Arithmetic.

3 Определения

Для целей данной Рекомендации | Международного стандарта используются следующие определения:

3.1 Термины ASN.1

В настоящей Рекомендации | Международном стандарте используются следующие термины, определенные в Рек. МСЭ-Т X.680 | ИСО/МЭК 8824-1:

a) тип choice;

b) тип sequence;

c) тип sequence-of.

3.2 Термины ECN

В настоящей Рекомендации | Международном стандарте используются следующие термины, определенные в Рек. МСЭ-Т X.692 | ИСО/МЭК 8825-3:

a) Модули определения кодирования (EDM);

b) Модуль связи кодирования (ELM).

3.3 Термины ИСО/МЭК 10646

В настоящей Рекомендации | Международном стандарте используются следующие термины, определенные в ИСО/МЭК 10646:

a) Основная многоязычная плоскость.

3.4 Дополнительные определения

3.4.1 Base64: Механизм кодирования, представляющий значение строки октетов как строку символов с использованием ограниченного алфавита, состоящего из 65 символов (см. п. 10.3 и IETF RFC 2045).

3.4.2 строка символов: Строка абстрактных символов ИСО/МЭК 10646 без какого-либо указания на то, каким способом они закодированы.

3.4.3 алгоритм кодирования: Точное определение того, как эффективно закодировать строку символов с определенными характеристиками в октеты.

ПРИМЕЧАНИЕ. – Примером является кодирование такой строки, как "-32176" в двоичное целое число с поразрядным дополнением до двух в двух октетах. 2-октетное кодирование сопровождается индексом словарной таблицы, идентифицирующим этот алгоритм кодирования.

3.4.4 внешний словарь: Набор словарных таблиц, на который ссылкой на который является URI (см. п. 7.2.14).

3.4.5 документ Быстрого информационного набора: Информационный набор XML, представленный как определено в настоящей Рекомендации | Международном стандарте.

3.4.6 окончательный словарь: Содержимое словарных таблиц по окончании создания или обработки документа Быстрого информационного набора.

3.4.7 единица информации: Каждый из типов единиц, составляющих Информационный набор XML.

3.4.8 исходный словарь: Набор словарных таблиц, установленных информацией, содержащейся в головной части документа Быстрого информационного набора, который (необязательно) ссылается на внешний словарь и (необязательно) предоставляет дополнительные записи таблицы.

3.4.9 заменитель имени: Набор из трех индексов словарной таблицы (первые два являются необязательными), которые используются для представления уточненного имени (см. п. 3.4.11).

3.4.10 документ XML, корректный с точки зрения пространств имен: Либо документ XML 1.0, сформированный корректно с точки зрения Пространств имен XML 1.0 W3C, либо документ XML 1.1, сформированный корректно с точки зрения Пространств имен XML 1.1 W3C.

3.4.11 **уточненное имя**: Набор, состоящий из свойств **[prefix]**, **[namespace name]**, и **[local name]** информационной единицы **element** или информационной единицы **attribute**.

3.4.12 **ограниченный алфавит**: Упорядоченный набор отдельных символов ИСО/МЭК 10646, который позволяет осуществлять компактное кодирование любой строки символов, полностью состоящей из символов, входящих в данный набор.

3.4.13 **индекс словарной таблицы**: Положительное целочисленное значение, идентифицирующее запись словарной таблицы.

3.4.14 **словарные таблицы**: Набор концептуальных таблиц (как правило, хотя и необязательно, создаваемых динамически), связанных с документом Быстрого информационного набора, которые содержат строки символов или другую информацию и поддерживают использование положительных значений типа "малое целое" (индексов словарных таблиц) для идентификации их записей.

ПРИМЕЧАНИЕ. – Примерами словарных таблиц являются таблицы, содержащие символьные строки, представляющие собой свойства **[local name]** единиц информации **attribute** или **element**, или строки символов, соответствующие последовательностям единиц информации **character,** являющимся членами свойства **[children]** единиц информации **element**.

3.4.15 **декларация XML**: Кодирование UTF-8 определенной строки символов (см. также п. 12.3), которое может быть приведено в начале документа Быстрого информационного набора для идентификации данного кодирования как документа Быстрого информационного набора и для того, чтобы отличить его от документов XML 1.0 W3C и документов XML 1.1 W3C.

3.4.16 **информационный набор XML**: Абстрактный набор данных, описывающий информацию в документе XML, корректном с точки зрения пространств имен, как определено в Информационном наборе XML W3C.

3.4.17 **пробельный символ XML**: Один или более символов Unicode HORIZONTAL TABULATION(горизонтальная табуляция) (9), LINE FEED(перевод строки) (10), CARRIAGE RETURN(возврат каретки) (13), or SPACE(пробел) (32).

ПРИМЕЧАНИЕ. – Эти символы – из числа тех, что соответствуют символам "S" (пробельные символы) как в XML 1.0 W3C, так и в XML 1.1 W3C (см. XML 1.0 W3C, п. 2.3 и XML 1.1 W3C, п. 2.3). Символы NEXT LINE (133) и LINE SEPARATOR (8232), которые могут встречаться в корректных с точки зрения пространств имен документах XML 1.1 W3C (см. XML 1.1 W3C, п. 2.11), преобразуются в символы LINE FEED средствами обработки конца строки (см. XML 1.1, п. 2.11). В случае если эти символы встречаются в Информационном наборе XML, созданном из корректного с точки зрения пространств имен документа XML 1.1 W3C, они не являются пробельными символами (whitespace) XML.

4 Сокращения

Для целей данной Рекомендации | Международного стандарта используются следующие сокращения:

ASN.1	Abstract Syntax Notation One	Абстрактная синтаксическая нотация версии 1
BMP	Basic Multilingual Plane	Основная многоязычная плоскость
ECN	Encoding Control Notation	Нотация контроля кодирования
MIME	Multipurpose Internet Mail Extensions	Многофункциональные расширения почтовой службы сети Internet
UBL	Universal Business Language	Универсальный язык бизнеса
URI	Uniform Resource Identifier	Унифицированный идентификатор ресурса
UTF-8	Universal Transformation Function 8-bit (see ISO/IEC 10646, Annex D)	Универсальная функция трансформации, 8 бит (см. ИСО/МЭК 10646, Приложение D)
UTF-16BE	Universal Transformation Function 16-bit Big Endian (see Unicode, 2.6)	Универсальная функция трансформации, 16 бит Big Endian (обратный порядок передачи байтов) (см. Unicode, п.2.6)
UUID	Universally Unique Identifier	Универсальный уникальный идентификатор
XML	eXtensible Markup Language	Расширяемый язык разметки

5 Нотация

5.1 В настоящей Рекомендации | Международном стандарте используется нотация ASN.1, определенная в Рек. МСЭ-Т X.680 | ИСО/МЭК 8824-1 для формального определения типов данных, кодирования которых представляют собой документы Быстрого информационного набора.

ПРИМЕЧАНИЕ. – В пункте 12 определяется применение Рек. МСЭ-Т X.692 | ИСО/МЭК 8825-3 к определению типов ASN.1, обеспечивающее кодирование битового уровня документа Быстрого информационного набора.

5.2 В настоящей Рекомендации | Международном стандарте **`полужирное начертание гарнитурой Courier`** используется для отображения нотации ASN.1, а **полужирное начертание гарнитурой Arial** используется для отображения синтаксиса XML W3C и для имен единиц информации Информационного набора XML.

5.3 Имена свойств единиц информации отображаются **полужирным начертанием гарнитурой Arial**, и заключаются в квадратные скобки (например, **[children]**).

5.4 Имена категорий строк символов (см. п. 8.4.2) и имена категорий уточненных имен отображаются в ВЕРХНЕМ РЕГИСТРЕ.

5.5 В настоящей Рекомендации | Международном стандарте позиции битов внутри октета определяются с использованием терминологии первый бит, второй бит и т.д. до восьмого бита, где первый бит является самым старшим битом октета, а восьмой бит является самым младшим битом октета.

6 Принципы построения и использования словарных таблиц

6.1 Словарные таблицы представляют собой концептуальные таблицы, устанавливающие соответствие между индексом словарной таблицы и записью словарной таблицы.

ПРИМЕЧАНИЕ. – Представление словарных таблиц в памяти компьютера не определяется, а также не определяются средства, при помощи которых в рамках конкретной реализации осуществляется установление соответствия между индексом словарной таблицы и записью словарной таблицы для этой таблицы.

6.2 Создатель документа Быстрого информационного набора из информационного набора XML определяет содержание словарных таблиц.

6.3 В самом общем случае в головной части документа Быстрого информационного набора может содержаться ссылка на набор словарных таблиц (внешний словарь), за которой следует определение дополнений к этим словарным таблицам для того, чтобы сформировать исходный словарь для документа Быстрого информационного набора. Последующие добавления в словарные таблицы имеют место в ходе создания и обработки документа Быстрого информационного набора, так, что они постепенно увеличиваются, формируя окончательный словарь для этого документа.

6.4 Некоторые словарные таблицы постепенно увеличиваются от состояния исходного словаря к окончательному словарю в ходе создания и обработки документа Быстрого информационного набора, и в связи с этим к имени таких словарных таблиц добавляется слово "динамическая". Механизмов удаления записей из каких-либо таблиц не предусмотрено.

6.5 Индексы словарных таблиц назначаются неявным образом. Первой записи в любой словарной таблице соответствует индекс словарной таблицы равный единице, и каждой последующей записи, вносимой в данную таблицу, присваивается следующее по порядку целочисленное значение индекса словарной таблицы. Там, где в настоящей Рекомендации | Международном стандарте определяется, что что-то должно быть добавлено в словарную таблицу, это подразумевает, что должен быть назначен следующий доступный индекс словарной таблицы.

ПРИМЕЧАНИЕ. – Значения индексов словарных таблиц начинаются с единицы, а не с нуля потому, что значение ноль, если его использование разрешено, имеет специальное значение "пустая строка символов" в поле, которое в другом случае могло бы содержать индекс словарной таблицы.

6.6 Для того, чтобы обеспечить поддержку такого неявного назначения индексов словарных таблиц, полностью определен концептуальный порядок обработки компонентов (на любой глубине) документа Быстрого информационного набора.

ПРИМЕЧАНИЕ. – Этот порядок совпадает с порядком кодирования компонентов в документе Быстрого информационного набора. Это не обязательно предполагает, что значение, которое содержит данные документа, обрабатывается в этом порядке. Данный порядок определен только для того, чтобы убедиться в том, что один и тот же индекс словарной таблицы присвоен любой заданной записи словарной таблицы как создателем, так и обработчиком документа Быстрого информационного набора.

6.7 Словарные таблицы применяются для многих целей (см. п. 8), однако их первоочередное назначение – предоставить возможность использования индексов словарных таблиц вместо записей словарных таблиц, если эти индексы меньше по размеру (и, возможно, могут быть быстрее обработаны), чем записи. Ряд встроенных записей для некоторых словарных таблиц определен в пункте 9. Эти записи всегда неявно присутствуют в этих словарных таблицах и имеют значения индексов словарных таблиц, определенные в п. 9.

6.8 Для некоторых категорий строк символов создатель документа Быстрого информационного набора имеет возможность выбрать добавлять или нет данную строку в словарную таблицу, основываясь на предполагаемой (или известной) частоте появления данной строки символов в этом Информационном наборе XML.

6.9 Точная форма и значение записей словарных таблиц определены в п. 8, однако в большинстве случаев они представляют собой строки символов переменной длины, часто короткие, однако потенциально их длина может достигать 2^{32} октета.

6.10 Соответствующий требованиям создатель документов Быстрого информационного набора должен осуществлять все добавления в словарные таблицы как определено в п. 7.13.7, п. 7.14.6, п. 7.14.7 и п. 7.16.7. Это позволяет обеспечить, чтобы количество записей в каждой словарной таблице не превышало 2^{20}.

ПРИМЕЧАНИЕ. – Запись словарной таблицы может быть идентична одной или нескольким другим записям словарной таблицы. Это позволяет эффективно создавать документы Быстрого информационного набора. Однако повторяющиеся записи уменьшают эффективность передачи. Повторяющиеся записи не влияют на работу обработчика.

6.11 Соответствующий требованиям обработчик документа Быстрого информационного набора должен осуществлять все добавления в словарные таблицы как определено в п. 7.13.8, п. 7.14.11, и п. 7.16.8. Этим обеспечивается соблюдение ограничений п. 6.10.

7 Определение типов ASN.1

7.1 Общие положения

7.1.1 В настоящей Рекомендации | Международном стандарте определяется набор типов ASN.1, поддерживающих представление Информационного набора XML. Корневым типом данного набора типов является тип **`Document`**.

7.1.2 На содержание Информационных наборов XML накладываются некоторые ограничения, а также представление несколько упрощено (см. п. 11) для того, чтобы повысить удобство использования данной спецификации и эффективность кодирований, получаемых с ее помощью.

ПРИМЕЧАНИЕ. – Информационный набор XML, не соответствующий данным ограничениям, не может быть представлен как в виде документа Быстрого информационного набора, так и в виде корректного с точки зрения пространств имен документа XML.

7.1.3 Для каждого вида единиц информации, определенного в Информационном наборе XML W3C, в настоящей Рекомендации | Международном стандарте приводится определение соответствующего типа ASN.1. Это определение типа всегда относится к типу sequence, причем компоненты соответствуют свойствам единицы информации.

7.1.4 Определенные свойства единиц информации не включены в определения типов ASN.1 (см. п. 11.4).

7.1.5 В некоторых случаях значение свойства, не включенного в определения типов ASN.1, может быть выведено из значений других свойств той же самой единицы информации или других единиц информации, включенных в определения типов ASN.1. В таких случаях опущение этого свойства упрощает представление без потерь информации. Однако существуют и случаи, когда значение не включенного 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 пункте 12 определяется кодирование типа the **`Document`**.

7.2 Тип `Document`

7.2.1 Тип **`Document`**:

```
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 }}
```

где значение **one-meg**:

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

Тип **NonEmptyOctetString**:

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

где значение **four-gig**:

```
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 **DocumentTypeDeclaration** определены в п. 7.17, п. 7.15, п. 7.11, п. 7.10, п. 7.14, п. 7.3, п. 7.5, п. 7.8, и п. 7.9 соответственно.

7.2.3 Тип **URI** должен представлять собой URI как определено в IETF RFC 2396.

7.2.4 Компонент **restricted-alphabets** в **initial-vocabulary** (если таковой присутствует) должен нести в себе одну или более строк символов, каждая их которых содержит символы ограниченного алфавита. Каждая строка символов должна содержать как минимум два символа, и все символы в данной строке символов должны быть различными.

ПРИМЕЧАНИЕ. – Использование ограниченного алфавита для оптимизации кодирования строк символов определено в п.7.17.6.

7.2.5 Компонент **encoding-algorithms** в **initial-vocabulary** (если таковой присутствует) должен нести в себе один или более URI, каждый из которых должен идентифицирует алгоритм кодирования.

ПРИМЕЧАНИЕ. – В настоящей Рекомендации | Международном стандарте определены встроенные алгоритмы кодирования (пункт 10) с определенными индексами словарных таблиц, однако определение дополнительных алгоритмов кодирования и связанных с ними индексов словарных таблиц, а также средств определения таких алгоритмов выходит за рамки настоящей Рекомендации | Международного стандарта. Информация, необходимая для определения алгоритма кодирования, определена в п. 8.3.3.

7.2.6 Тип **Document**представляет единицу информации **document** Информационного набора XML. Поскольку все остальные единицы информации в Информационном наборе XML либо свойствами этой единицы информации, либо свойствами единиц, являющихся дочерними элементами или потомками данной единицы информации (любой глубины), каждая единица информации **Document** представляет Информационный набор XML целиком.

ПРИМЕЧАНИЕ. – Каждая единица информации **Document** без ссылки на внешний словарь (см. п. 7.2.13) также определяет окончательный словарь, который может быть использован как внешний словарь для какого-либо другого документа Быстрого информационного набора.

7.2.7 Компонент **additional-data** (если таковой присутствует) должен нести в себе один или более компонентов **additional-datum** для того, чтобы сделать возможным использование дополнительных механизмов для обработки документов Быстрого информационного набора.

ПРИМЕЧАНИЕ 1. – Примером являются данные, которые дают обработчику возможность осуществлять доступ к частям документа Быстрого информационного набора без необходимости обрабатывать документ целиком. Форма такого рода данных не стандартизирована.

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

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

a) Компонента **id** (значение типа **URI**); URI должен являться ссылкой на спецификацию определяющую форму и семантику компонента, **data**;

ПРИМЕЧАНИЕ. – Форма **additional-datum** может быть определена как абстрактный тип, объединенный с правилом кодирования или при помощи каких-либо других подходящих средств.

b) Компонент **data**, представляющий собой строку октетов, содержащую дополнительные данные обработки.

7.2.9 Компонент **additional-data** должен использоваться в соответствии со следующими правилами:

a) Компонент **additional-datum** может быть проигнорирован обработчиком во всех случаях, кроме тех, когда URI распознан и дополнительная обработка признана важной для деятельности обработчика;

b) Обработчик, игнорирующий все компоненты **additional-datum**, тем не менее имеет возможность создания Информационного набора XML эквивалентного Информационному набору XML, использованному для создания данного документа Быстрого информационного набора.

7.2.10 Допускается присутствие нескольких компонентов **additional-datum** с одним и тем же URI, такие компоненты обрабатываются в соответствии со спецификацией, связанной с этим URI.

7.2.11 Компонент **initial-vocabulary** предоставляет данные, которые (совместно с некоторыми встроенными записями таблиц) полностью определяют первоначальное содержание таблицы ограниченных алфавитов (см. п. 8.2), таблицы алгоритмов кодирования (см. п. 8.3), динамических таблиц строк (см. п. 8.4) и динамических таблиц имен (см. п. 8.5) данного документа Быстрого информационного набора (первоначальный словарь документа Быстрого информационного набора). Первоначальный словарь состоит из следующих данных:

a) упорядоченный набор ограниченных алфавитов (см. п. 8.2.2), содержащий как минимум встроенные ограниченные алфавиты (см. пункт 9);

b) упорядоченный набор алгоритмов кодирования (см. п. 8.2.2), содержащий как минимум встроенные алгоритмы кодирования (см. пункт 10);

c) восемь независимых упорядоченных наборов строк символов, соответствующих восьми категориям строк символов, определенным в настоящей Рекомендации | Международном стандарте (см. п. 8.4.2), причем каждый из наборов содержит ноль или более строк символов только для одной категории;

d) два независимых упорядоченных набора заменителей имен (см. п. 8.5.2), соответствующих двум категориям уточненных имен, определенным в настоящей Рекомендации | Международном стандарте (см. п. 8.5.4), причем каждый из наборов содержит ноль или более заменителей имен только для одной категории.

ПРИМЕЧАНИЕ. – Исходный словарь не может быть полностью пустым, поскольку он всегда содержит (как минимум) встроенные ограниченные алфавиты и встроенные алгоритмы кодирования. Однако для документов Быстрого информационного набора не является необычной ситуация, когда исходный словарь содержит только эти данные, поскольку решение (принимаемое создателем документа Быстрого информационного набора) о том, как использовать компонент **initial-vocabulary** зависит от конкретной реализации, и в рамках некоторых реализаций может быть решено добавлять записи в словарные таблицы динамически (внутри тела документа Быстрого информационного набора).

7.2.12 Исходный словарь документа Быстрого информационного набора должен определяться следующим образом:

a) Если компонент **initial-vocabulary** отсутствует, то исходный словарь должен состоять только из встроенных записей таблиц, определенных в п. 7.2.21, п. 7.2.22 и в пунктах 9 и 10.

b) Если компонент **initial-vocabulary** присутствует, и компонент **external-vocabulary** отсутствует, то исходный словарь должен состоять из встроенных записей таблиц, определенных в п. 7.2.21, п. 7.2.22 и в пунктах 9 и 10, совместно с добавочными записями таблиц (если таковые имеются), определенными согласно п. 7.2.16.

c) Если компонент **initial-vocabulary** и компонент **external-vocabulary** присутствует, то исходный словарь должен состоять из окончательного словаря, определенного компонентом **external-vocabulary**, как определено в п. 7.2.13 и п. 7.2.14, совместно с добавочными записями таблиц (если таковые имеются), определенными согласно п. 7.2.16.

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

ПРИМЕЧАНИЕ. – В настоящей Рекомендации | Международном стандарте не определяются никакие внешние словари и никакие URI, который являются ссылками на внешние словари. Такие внешние словари и URI могут быть определены компетентными органами, имеющими возможность выделять такие URI и могут быть как согласованы в частном порядке, так и подлежать стандартизации.

7.2.14 Внешний словарь может быть определен одним из трех способов:

a) как окончательный словарь документа Быстрого информационного набора, который сам по себе не должен ссылаться на внешний словарь;

ПРИМЕЧАНИЕ 1. – Хранится ли окончательный словарь локально, или хранится только документ Быстрого информационного набора, а окончательный словарь создается динамически путем его обработки, зависит от конкретной реализации.

ПРИМЕЧАНИЕ 2. – Ограничение, состоящее в том, что окончательный словарь документа Быстрого информационного набора со ссылкой на внешний словарь на может сам быть использован в качестве внешнего словаря, налагается для того, чтобы упросить реализацию и избежать появления циклических ссылок.

b) как корректный с точки зрения пространств имен документ XML, который концептуально обрабатывается следующим образом:

1) должен быть определен Информационный набор XML для данного корректного с точки зрения пространств имен документа XML;

2) для данного информационного набора XML должен быть создан документ Быстрого информационного набора как определено в настоящей Рекомендации | Международном стандарте, но он не должен содержать компонента **initial-vocabulary**, а для компонента **add-to-table** типа **NonIdentifyingStringOrIndex** (см. п. 7.14) должно всегда быть установлено значение **TRUE** и ни в одной из таблиц строк символов не должно присутствовать нескольких идентичных строк символов;

3) окончательный словарь этого документа Быстрого информационного набора становится внешним словарем;

ПРИМЕЧАНИЕ 3. – Хранится ли окончательный словарь локально, или хранится только документ XML, а окончательный словарь создается путем его обработки, зависит от конкретной реализации.

c) как набор словарных таблиц, определенных с использованием любого другого в достаточной мере точного механизма или текста, который должен включать встроенные записи таблиц из пунктов 9 и 10 (с индексами словарных таблиц, определенными в упомянутых пунктах).

ПРИМЕЧАНИЕ 4. – Определение нотации для определения словарных таблиц выходит за рамки настоящей Рекомендации | Международного стандарта.

ПРИМЕЧАНИЕ 5. – Установленное для случаев использования этого механизма требование об обязательном включении встроенных записей словарных таблиц позволяет удостовериться, что все словарные таблицы содержат встроенные записи таблиц.

7.2.15 Для внешнего словаря, определенного в соответствии с п. 7.2.14, всем записям таблицы, содержащим строки и имена, за исключением записей в таблицах 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 ограниченным алфавитам, 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`**, и **`NameSurrogate`** (если таковые присутствуют), который присутствует в любом из оставшихся компонентов **`initial-vocabulary`**, должен быть добавлен по порядку (см. п. 8.1) в словарную таблицу, как определено в Таблице 1.

Таблица 1 – Соответствие идентификаторов компонентов словарным таблицам

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

7.2.17 Значение типа **`NonEmptyOctetString`** должно нести в себе кодирование UTF-8 (см. ИСО/МЭК 10646, Приложение D) строки символов.

7.2.18 Таблица ограниченных алфавитов и таблица алгоритмов кодирования в исходном словаре должны содержать максимум 256 записей. Все остальные таблицы должны содержать максимум 2^{20} записей.

ПРИМЕЧАНИЕ. – Ограничение на количество записей накладывается с тем, чтобы удостовериться в соблюдении общих верхних границ для индексов таблиц. Данное ограничение также действует если записи таблицы добавляются динамически (см. п. 7.13.7, п. 7.14.6, п. 7.14.7 и п. 7.16.7). Данные ограничения не препятствуют кодированию любого Информационного набора XML как документа Быстрого информационного набора.

7.2.19 Встроенные ограниченные алфавиты имеют индексы словарных статей в диапазоне от 1 до 2 (см. пункт 9). Индексы словарных таблиц ограниченных алфавитов в компоненте **`restricted-alphabets`** в **`initial-vocabulary`** (если таковой присутствует) должны назначаться следующим образом:

a) если нет внешнего словаря, или внешний словарь содержит только встроенные ограниченные алфавиты, то индексы назначаются начиная с 16;

b) в остальных случаях индексы должны назначаться начиная с единицы плюс наибольший индекс ограниченного алфавита во внешнем словаре.

ПРИМЕЧАНИЕ. – Это означает, что индексы словарных таблиц от 3 до 15 не используются. Эти значения зарезервированы для последующих версий настоящей Рекомендации | Международного стандарта.

7.2.20 Встроенные алгоритмы кодирования имеют индексы словарных статей в диапазоне от 1 до 10 (см. пункт 10). Индексы словарных таблиц алгоритмов кодирования в компоненте **`encoding-algorithms`** в **`initial-vocabulary`** (если таковой присутствует) должны назначаться следующим образом:

a) если нет внешнего словаря, или внешний словарь содержит только встроенные алгоритмы кодирования, то индексы назначаются начиная с 32;

b) в остальных случаях индексы должны назначаться начиная с единицы плюс наибольший индекс алгоритма кодирования во внешнем словаре.

ПРИМЕЧАНИЕ. – Это означает, что индексы словарных таблиц от 11 до 31 не используются. Эти значения зарезервированы для последующих версий настоящей Рекомендации | Международного стандарта.

7.2.21 Таблица PREFIX должна содержать встроенную запись префикса (prefix) "xml", которой должен быть назначен индекс 1. Индексы словарных таблиц префиксов в компоненте **`prefixes`** в **`initial-vocabulary`** (если таковой присутствует) должны назначаться следующим образом:

a) если нет внешнего словаря, или внешний словарь содержит только встроенную запись префикса, то индексы назначаются начиная с 2;

b) в остальных случаях индексы должны назначаться начиная с единицы плюс наибольший индекс префикса во внешнем словаре.

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

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

Данной записи должен быть присвоен индекс со значением 1.

7.2.23 Индексы словарных таблиц пространств имен в компоненте **`namespace-names`** в **`initial-vocabulary`** (если таковой присутствует) должны назначаться следующий образом:

a) если нет внешнего словаря, или внешний словарь содержит только встроенную запись пространства имен, то индексы назначаются начиная с 2;

b) в остальных случаях индексы должны назначаться начиная с единицы плюс наибольший индекс пространства имен во внешнем словаре.

7.2.24 Компонент **`notations`** представляет свойство **[notations]** единицы информации **document.** Тип данного компонента - sequence-of, хотя свойство **[notations]** и определено в Информационном наборе XML W3C как unordered set (неупорядоченный набор) (единиц информации **notation**).

ПРИМЕЧАНИЕ. – Здесь и далее, используется тип sequence-of , а не тип set-of, поскольку последний не удовлетворяет потребности в строгом упорядочении всех компонентов документа Быстрого информационного набора (см. п. 8.1).

7.2.25 Компонент **`unparsed-entities`** представляет свойство **[unparsed entities]** единицы информации **document**. Тип данного компонента - sequence-of, хотя свойство **[unparsed entities]** и определено в Информационном наборе XML W3C как unordered set (неупорядоченный набор) (единиц информации **unparsed entity**).

7.2.26 Компонент **`character-encoding-scheme`** представляет свойство **[character encoding scheme]** единицы информации document. Данный компонент относится к типу **`NonEmptyOctetString`** и значение данного компонента должно нести в себе кодирование UTF-8 (см. ИСО/МЭК 10646, Приложение D) свойства **[character encoding scheme]**. Отсутствие этого компонента в абстрактном значении типа **`Document`** указывает, что свойство **[character encoding scheme]** имеет значение "UTF-8".

ПРИМЕЧАНИЕ. – Поддержка свойства **[character encoding scheme]** преобразование документов XML в документы Быстрого информационного набора без изменения схемы кодирования символов. Создатель документа Быстрого информационного набора может закодировать свойство **[character encoding scheme]** полученное из декларации кодирования документа XML (см. XML 1.0 W3C, п. 4.3.1 и XML 1.1 W3C, п. 4.3.1). Обработчик документа Быстрого информационного набора может использовать компонент **`character-encoding-scheme`** (если таковой присутствует) если он желает воспроизвести оригинальное кодирование.

7.2.27 Компонент **`standalone`** представляет свойство **[standalone]** единицы информации **document**. Абстрактное значение **`TRUE`** представляет значение данного свойства **yes**, а абстрактное значение **`FALSE`** представляет значение **no**. Отсутствие данного компонента в абстрактном значении типа **`Document`** указывает, что у свойства **[standalone]** нет значения.

7.2.28 Компонент **`version`** представляет свойство **[version]** единицы информации **document**. Данный компонент относится к типу **`NonIdentifyingStringOrIndex`** (см. п. 7.14), представляя здесь строку символов категории OTHER STRING. Отсутствие данного компонента в абстрактном значении типа **`Document`** указывает, что у свойства **[version]** нет значения.

7.2.29 Компонент **`children`** представляет свойство **[children]** единицы информации **document**. Ровно одна единица типа sequence-of (в любой позиции) должна использовать альтернативный **`element`** типа choice, и не более чем одна из единиц (в любой позиции) должна использовать альтернативный **`document-type-declaration`**. Все остальные единицы (если таковые присутствуют) должны использовать либо альтернативный **`processing-instruction`** либо альтернативный **`comment`**.

7.2.30 Свойство **[document element]** единицы информации **document** не включается в тип **`Document`**. Значением данного свойства всегда является одна и единственная единица информации **element**, являющаяся членом свойства **[children]** единицы информации **document**.

7.2.31 Свойство **[base URI]** единицы информации **document** не включается в тип **Document** и не поддерживается в рамках настоящей Рекомендации | Международного стандарта.

7.2.32 Свойство **[all declarations processed]** единицы информации **document** не включается в тип **Document**, и принимается имеющим значение **true** (см. п. 11.3).

7.3 Тип Element

7.3.1 Тип **Element**:

```
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 п.7.12, п.7.16, п.7.4, п.7.5, п.7.6, п.7.7, и п.7.8 соответственно.

7.3.3 Тип **Element** представляет единицу информации **element** Информационного набора XML.

7.3.4 Компонент **namespace-attributes** представляет свойство **[namespace attributes]** единицы информации **element**. Тип данного компонента - sequence-of, хотя свойство **[namespace attributes]** и определено в Информационном наборе XML W3C как unordered set (неупорядоченный набор) (единиц информации **attribute**).

ПРИМЕЧАНИЕ. – Тип данного компонента sequence-of – **NamespaceAttribute**, а не **Attribute**, хотя свойство **[namespace attributes]** единицы информации **element** и определено в Информационном наборе XML W3C как набор единиц информации **attribute**. В ограниченном Информационном наборе XML (см. п. 11.3), свойства единицы информации **namespace** могут быть выведены из свойств единицы информации **attribute** представляющей атрибут пространства имен. Существует только частичная возможность выведения свойств в обратном направлении, однако это ограничение признается приемлемым для предполагаемых вариантов использования настоящей Рекомендации | Международного стандарта (см. также примечание в п. 7.2.24).

7.3.5 Компонент **qualified-name** представляет уточненное имя (см. п. 3.4.11) единицы информации **element** (т.е. набор, состоящий из свойств **[prefix]**, **[namespace name]**, и **[local name]** этой единицы информации). Данный компонент относится к типу **QualifiedNameOrIndex** (см. п. 7.16), представляющему здесь уточненное имя категории ELEMENT NAME.

7.3.6 Компонент **attributes** представляет свойство **[attributes]** единицы информации **element.** Тип данного компонента - sequence-of, хотя свойство **[attributes]** и определено в Информационном наборе XML W3C как unordered set (неупорядоченный набор) (единиц информации **attribute**).

7.3.7 Компонент **children** представляет свойство **[children]** единицы информации **element.** Когда два или более соседних дочерних элемента являются единицами информации **character,** одна единица **CharacterChunk** может быть использована для представления этих единиц информации **character**.

ПРИМЕЧАНИЕ. – Если существует последовательность из N соседних символов среди дочерних элементов единицы информации **element**, то позволена любая группировка этих N символов в серию последовательных блоков символов. Однако предполагается, что создатель документа Быстрого информационного набора сделает каждый из блоков символов настолько большим, насколько это будет возможно, чтобы создать эффективные кодировки.

7.3.8 Свойство **[in-scope namespaces]** единицы информации **element** не включается в тип **Element**.

ПРИМЕЧАНИЕ. – В ограниченном Информационном наборе XML (см. п. 11.3), свойство **[in-scope namespaces]** единицы информации **element** может быть выведено из свойства **[namespace attributes]** единицы информации **element** вместе со свойством **[namespace attributes]** всех единиц информации **element** (если таковые имеются) содержащих (прямо или косвенно) эту единицу информации **element**.

7.3.9 Свойство **[base URI]** единицы информации **element** не включается в тип **Element** и не поддерживается в рамках настоящей Рекомендации | Международного стандарта.

7.3.10 Свойство **[parent]** единицы информации **element** не включается в тип **Element**. Значением данного свойства для любой заданной единицы информации **element** является единица информации **document** или **element**, содержащую данную единицу информации как члена своего свойства **[children]**.

7.4 Тип Attribute

7.4.1 Тип **Attribute**:

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

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

7.4.3 Тип **Attribute** представляет единицу информации **attribute** Информационного набора XML.

7.4.4 Компонент **qualified-name** представляет уточненное имя (см. п. 3.4.11) единицы информации **attribute** (представляющей собой набор из свойств **[prefix], [namespace name]**, и **[local name]** этой единицы информации). Данный компонент относится к типу **QualifiedNameOrIndex** (см. п. 7.16), представляющему здесь уточненное имя категории ATTRIBUTE NAME.

7.4.5 Компонент **normalized-value** представляет свойство **[normalized value]** единицы информации **attribute**. Данный компонент относится к типу **NonIdentifyingStringOrIndex** (см. 7.14), представляющему здесь уточненное имя категории ATTRIBUTE VALUE.

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

ПРИМЕЧАНИЕ. – Данное ограничение накладывается определением ASN.1, которая разработана для оптимизации кодирований и для простоты реализации (см. также п. 11.3 j).

7.4.7 Свойство **[specified]** единицы информации **attribute** не включается в тип **Attribute**.

7.4.8 Свойство **[attribute type]** единицы информации **attribute** не включается в тип **Attribute**.

7.4.9 Свойство **[references]** единицы информации **attribute** не включается в тип **Attribute**.

ПРИМЕЧАНИЕ. – В ограниченном Информационном наборе XML (см. п. 11.3) свойство **[references]** единицы информации **attribute** может быть выведено из свойства **[normalized value]** единицы информации **attribute** вместе со свойствами других единиц информации в Информационном множестве XML.

7.4.10 Свойство **[owner element]** единицы информации **attribute** не включается в тип **Attribute**. Значение этого свойства для любой заданной единицы информации **attribute** представляет собой единицу информации **element**, содержащую эту единицу информации в качестве члена свойства **[attributes]**.

7.5 Тип ProcessingInstruction

7.5.1 Тип **ProcessingInstruction**:

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

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

7.5.3 Тип **ProcessingInstruction** представляет единицу информации **processing instruction** Информационного набора XML.

7.5.4 Компонент **target** представляет свойство **[target]** единицы информации **processing instruction**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER NCNAME.

7.5.5 Компонент **content** представляет свойство **[content]** единицы информации **processing instruction**. Данный компонент относится к типу **NonIdentifyingStringOrIndex** (см. п. 7.14), представляющему здесь строку символов категории OTHER STRING.

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

ПРИМЕЧАНИЕ. – Данное ограничение накладывается определением ASN.1, которая разработана для оптимизации кодирований и для простоты реализации (см. также п. 11.3 j).

7.5.7 Свойство **[notation]** единицы информации **processing instruction** не включается в тип **`ProcessingInstruction`**.

ПРИМЕЧАНИЕ. – В ограниченном Информационном наборе XML (см. п. 11.3) свойство **[notation]** единицы информации **processing instruction** может быть выведено из свойства **[target]** единицы информации **processing instruction** вместе со свойством **[notations]** единицы информации **document**.

7.5.8 Свойство **[parent]** единицы информации **processing instruction** не включается в тип **`ProcessingInstruction`**. Значение этого свойства для любой заданной единицы информации **processing instruction** представляет собой единицу информации **document**, **element,** или **document type definition**, содержащую эту единицу информации в качестве члена свойства **[children]**.

7.6 Тип `UnexpandedEntityReference`

7.6.1 Тип **`UnexpandedEntityReference`**:

```
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 п. 7.13.

7.6.3 Тип **`UnexpandedEntityReference`** представляет единицу информации **unexpanded entity reference** Информационного набора XML.

7.6.4 Компонент **`name`** представляет свойство **[name]** единицы информации **unexpanded entity reference** Данный компонент относится к типу **`IdentifyingStringOrIndex`** (see 7.13), представляющему здесь строку символов категории OTHER NCNAME.

7.6.5 Компонент **`system-identifier`** представляет свойство **[system identifier]** единицы информации **unexpanded entity reference** Данный компонент относится к типу **`IdentifyingStringOrIndex`** (см п. 7.13), представляющему здесь строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении **`UnexpandedEntityReference`** указывает, что свойство **[system identifier]** не имеет значения.

7.6.6 Компонент **`public-identifier`** представляет свойство **[public identifier]** единицы информации **unexpanded entity reference** Данный компонент относится к типу **`IdentifyingStringOrIndex`** (см п. 7.13), представляющему здесь строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении **`UnexpandedEntityReference`** указывает, что у свойства **[public identifier]** нет значения.

7.6.7 Свойство **[declaration base URI]** единицы информации **unexpanded entity reference** не включается в тип **`UnexpandedEntityReference`** и не поддерживается в рамках настоящей Рекомендации | Международного стандарта.

7.6.8 Свойство **[parent]** единицы информации **unexpanded entity reference** не включается в тип **`UnexpandedEntityReference`**. Значение данного свойства для любой заданной единицы информации **unexpanded entity reference** представляет собой единицу информации **element**, содержащую эту единицу информации в качестве члена свойства **[children]**.

7.7 Тип `CharacterChunk`

7.7.1 Тип **`CharacterChunk`**:

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

7.7.2 Тип **`NonIdentifyingStringOrIndex`** определен в п. 7.14.

7.7.3 Тип **`CharacterChunk`** соответствует единице информации **character,** однако представляет ряд соседних единиц информации **character** (членов свойства **[children]** родительской единицы информации **element**), а не отдельную единицу информации **character**.

7.7.4 Количество единиц информации **character**, представляемых значением типа **`CharacterChunk`**, не должно быть равным нулю.

7.7.5 Компонент **`character-codes`** представляет свойство **[character code]** единицы (единиц) информации **character** в блоке. Данный компонент относится к типу **`NonIdentifyingStringOrIndex`** (см. п. 7.14), представляющему здесь строку символов категории CONTENT CHARACTER CHUNK.

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

ПРИМЕЧАНИЕ. – Данное ограничение накладывается определением ASN.1, которая разработана для оптимизации кодирований и для простоты реализации. Данное ограничение не делает невозможным кодирование единицы информации **element**, содержащей более чем 2^{32} единиц информации **characte**, поскольку возможно использование множественных блоков.

7.7.7 Свойство **[element content whitespace]** единицы (единиц) информации **character** не включается в тип **CharacterChunk**.

7.7.8 Свойство **[parent]** единицы (единиц) информации **character** не включается в тип **CharacterChunk**. . Значение данного свойства для любой заданной единицы информации **character** представляет собой единицу информации **element**, содержащую эту единицу информации в качестве члена свойства **[children]**.

7.8 Тип Comment

7.8.1 Тип **Comment**:

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

7.8.2 Тип **NonIdentifyingStringOrIndex** определен в п. 7.14.

7.8.3 Тип **Comment** представляет единицу информации **comment** Информационного набора XML.

7.8.4 Компонент **content** представляет свойство **[content]** единицы информации **comment**. Данный компонент относится к типу **NonIdentifyingStringOrIndex** (см. п. 7.14), представляющему здесь строку символов категории OTHER STRING.

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

ПРИМЕЧАНИЕ. – Данное ограничение накладывается определением ASN.1, которая разработана для оптимизации кодирований и для простоты реализации (см. также п. 11.3 j).

7.8.6 Свойство **[parent]** единицы информации **comment** не включается в тип **Comment**. Значение этого свойства для любой заданной единицы информации **comment** представляет собой единицу информации **document** или **element,** содержащую эту единицу информации в качестве члена свойства **[children]**.

7.9 Тип DocumentTypeDeclaration

7.9.1 Тип **DocumentTypeDeclaration**:

```
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 п. 7.13.

7.9.3 Тип **DocumentTypeDeclaration** представляет единицу информации **document type declaration** Информационного набора XML.

7.9.4 Компонент **system-identifier** представляет свойство **[system identifier]** единицы информации **document type declaration**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **DocumentTypeDeclaration** указывает, что свойство **[system identifier]** не имеет значения.

7.9.5 Компонент **public-identifier** представляет свойство **[public identifier]** единицы информации **document type declaration**. Данное свойство относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **DocumentTypeDeclaration** указывает, что свойство **[public identifier]** не имеет значения.

7.9.6 Компонент **children** представляет свойство **[children]** единицы информации **document type declaration**.

7.9.7 Свойство **[parent]** единицы информации **document type declaration** не включается в тип **DocumentTypeDeclaration**. Значение этого свойства для любой заданной единицы информации **document type declaration** представляет собой единицу информации **document,** содержащую эту единицу информации в качестве члена свойства **[children]**.

7.10 Тип UnparsedEntity

7.10.1 Тип **UnparsedEntity**:

```
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 п. 7.13.

7.10.3 Тип **UnparsedEntity** представляет единицу информации **unparsed entity** Информационного набора XML.

7.10.4 Компонент **name** представляет свойство **[name]** единицы информации **unparsed entity**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER NCNAME.

7.10.5 Компонент **system-identifier** представляет свойство **[system identifier]** единицы информации **unparsed entity**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER URI.

7.10.6 Компонент **public-identifier** представляет свойство **[public identifier]** единицы информации **unparsed entity**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **UnparsedEntity** указывает, что свойство **[public identifier]** не имеет значения.

7.10.7 Компонент **notation-name** представляет свойство **[notation name]** единицы информации **unparsed entity**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER NCNAME.

7.10.8 Свойство **[declaration base URI]** единицы информации **unparsed entity** не включается в тип **UnparsedEntity** и не поддерживается в рамках настоящей Рекомендации | Международного стандарта.

7.10.9 Свойство **[notation]** единицы информации **unparsed entity** не включается в тип **UnparsedEntity**.

ПРИМЕЧАНИЕ. – В ограниченном Информационном наборе XML (см. п. 11.3) свойство **[notation]** единицы информации **unparsed entity** может быть выведено из свойства **[notation name]** единицы информации **unparsed entity** вместе со свойством **[notations]** единицы информации **document**.

7.11 Тип Notation

7.11.1 Тип **Notation**:

```
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 п. 7.13.

7.11.3 Тип **Notation** представляет единицу информации **notation** Информационного набора XML.

7.11.4 Компонент **name** представляет свойство **[name]** единицы информации **notation**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER NCNAME.

7.11.5 Компонент **system-identifier** представляет свойство **[system identifier]** единицы информации **notation**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **Notation** указывает, что свойство **[system identifier]** не имеет значения.

7.11.6 Компонент **public-identifier** представляет свойство **[public identifier]** единицы информации **notation**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории OTHER URI. Отсутствие данного компонента в абстрактном значении типа **Notation** указывает, что свойство **[public identifier]** не имеет значения.

7.11.7 Свойство **[declaration base URI]** единицы информации **notation** не включается в тип **Notation** и не поддерживается в рамках настоящей Рекомендации | Международного стандарта.

7.12 Тип NamespaceAttribute

7.12.1 Тип **NamespaceAttribute**:

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

7.12.2 Тип **IdentifyingStringOrIndex** определен в п. 7.13.

7.12.3 Тип **NamespaceAttribute** представляет единицу информации **attribute**, являющуюся членом свойства **[namespace attributes]** единицы информации **element** Информационного набора XML.

ПРИМЕЧАНИЕ. – В Информационном наборе XML как атрибуты, так и атрибуты пространств имен являются единицами информации **attribute**. В рамках настоящей Рекомендации | Международного стандарта в целях оптимизации используются различные типы.

7.12.4 В Информационном наборе XML существует два типа атрибутов пространств имен:

a) декларации пространств имен по умолчанию: свойство **[prefix]** единицы информации **attribute** не имеет значения, а значение свойства **[local name]** – "**xmlns**";

b) декларации пространств имен, отличные от деклараций по умолчанию: свойство **[prefix]** единицы информации **attribute** имеет значение "**xmlns**", а свойство **[local name]** предоставляет префикс декларации пространств имен.

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

7.12.5 Если атрибут пространства имен – декларация пространства имен по умолчанию (случай a в п. 7.12.4), то компонент **prefix** должен отсутствовать, в противном случае (случай b в п. 7.12.4) он должен присутствовать и представлять свойство **[local name]** единицы информации **attribute**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории PREFIX.

7.12.6 Если свойство **[normalized value]** единицы информации **attribute** представляет собой пустую строку, то компонент **namespace-name** должен отсутствовать; в противном случае он должен присутствовать, представляя свойство **[normalized value]** единицы информации **attribute**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории NAMESPACE NAME.

7.12.7 Свойство **[namespace name]** единицы информации **attribute** всегда имеет значение "**http://www.w3.org/2000/xmlns/**" (см. Информационный набор XML W3C) и не включается в тип **NamespaceAttribute**.

7.13 Тип IdentifyingStringOrIndex

7.13.1 Тип **IdentifyingStringOrIndex**:

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

7.13.2 Тип **NonEmptyOctetString** и значение **one-meg** определены в п. 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) как значение типа **NonEmptyOctetString**, либо индекс словарной таблицы строки символов (заданной категории) в словарной таблице для этой категории строк (см. п. 8.4.2), которая называется соответствующей таблицей строк.

ПРИМЕЧАНИЕ 1. – Категория строки всегда определяется в связанном тексте предыдущих подпунктов (с п. 7.5 по п. 7.12), всякий раз, когда используется данный тип.

ПРИМЕЧАНИЕ 2. – "Идентифицирующие" и "не идентифицирующие" строки символов рассматриваются как различные строки. "Не идентифицирующие" строки символов могут быть закодированы одним из многих форматов кодирования, в то время как все "идентифицирующие" строки символов должны быть закодированы UTF-8. Кроме этого, "не идентифицирующие" строки символов могут как добавляться (на усмотрение создателя) в динамические таблицы строк, так и не добавляться в эти таблицы (см. п. 7.14.6), тогда как "идентифицирующие" строки символов всегда добавляются в динамические таблицы строк (см. п. 7.13.7).

7.13.5 Компонент **literal-character-string**, если таковой присутствует, должен содержать кодирование UTF-8 (см. ИСО/МЭК 10646, Приложение D) строки символов (см. п. 7.13.4).

7.13.6 Компонент **string-index**, если таковой присутствует, должен содержать индекс словарной таблицы любой из записей, идентичных данной строке символов, из соответствующей таблицы строк.

7.13.7 При создании абстрактного значения этого типа ASN.1 (представляющего заданную строку символов заданной категории), если идентичная строка символов существует в текущем содержимом соответствующей таблицы строк, создатель должен действовать либо по согласно варианту a), либо согласно варианту b), приведенным ниже, в зависимости от конкретной реализации (однако если возможно, следует выбирать первый вариант, поскольку он порождает наименьшее количество индексов, указывающих на одну и ту же строку символов), в противном случае (не существует идентичной строки символов) создатель должен действовать согласно варианту b). Варианты действий a) и b) приведены ниже:

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

b) выбрать вариант **literal-character-string**, назначить заданную строку символов **literal-character-string**, и добавить в соответствующую таблицу строк идентичную строку символов, за исключением случаев, когда данная таблица уже содержит 2^{20} записей.

ПРИМЕЧАНИЕ. – Результатом выбора варианта b) будет появление более чем одной идентичной строки символов в таблице строк (если таблица еще не содержит 2^{20} записей). Это не сказывается на последующей обработке строк символов (см. п. 7.13.8).

7.13.8 При обработке абстрактного значения этого типа ASN.1, представляющего строку символов (заданной категории), обработчик должен определить строку символов, представляемую данным абстрактным значением, следующим образом:

a) Если выбран вариант **string-index**, то строка символов, представляемая данным абстрактным значением, должна являться строкой символов в текущем содержимом соответствующей таблицы строк, индекс словарной таблицы для которой равен значению **string-index**.

b) Если выбран вариант **literal-character-string**, то строка символов, представляемая данным абстрактным значением, должна представлять собой значение **literal-character-string**, и идентичная строка символов должна быть добавлена в соответствующую таблицу строк (см. п. 7.13.9), за исключением случаев, когда данная таблица уже содержит 2^{20} записей.

ПРИМЕЧАНИЕ. – Результатом выбора варианта b) будет появление более чем одной идентичной строки символов в таблице строк (если таблица еще не содержит 2^{20} записей). Это не сказывается на последующей обработке строк символов (см. п. 7.13.8).

7.13.9 Если обработчик по какой-либо причине, включая ограничения, связанные с конкретной реализацией, не может добавить строку в словарную таблицу, содержащую менее чем 2^{20} записей в случае, когда такое добавление необходимо для выполнения действий согласно п. 7.13.8 b, он должен прекратить обработку документа Быстрого информационного набора и объявить об ошибке.

7.14 Тип NonIdentifyingStringOrIndex

7.14.1 Тип **NonIdentifyingStringOrIndex**:

```
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 п. 7.17 и п. 7.2.1 соответственно.

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

ПРИМЕЧАНИЕ. – Примером такой строки символов служит значение атрибута.

7.14.4 Абстрактное значение типа **NonIdentifyingStringOrIndex** содержит либо строку символов (заданной категории) как значение типа **EncodedCharacterString** (см. п. 7.17), либо индекс словарной таблицы строки символов заданной категории в словарной таблице для этой категории строк (см. п. 8.4.2), называемой соответствующей таблицей строк.

ПРИМЕЧАНИЕ. – Категория строки всегда определена в связанном тексте предыдущих пунктов, когда используется этот тип.

7.14.5 Компонент **string-index**, если таковой присутствует, должен либо иметь значение, равное нулю (обозначая строку символов нулевой длины – см. п. 7.14.6), либо содержать индекс словарной таблицы любой из записей соответствующей таблицы строк, идентичных данной строке символов.

7.14.6 Для строки символов нулевой длины создатель всегда должен использовать вариант **string-index** с целочисленным значением, равным нулю. Обработчик должен рассматривать такое значение как представляющее строку символов нулевой длины.

7.14.7 При создании абстрактного значения типа **NonIdentifyingStringOrIndex** (представляющего заданную строку символов заданной категории) ненулевой длины, если идентичная строка символов существует в текущем содержимом соответствующей таблицы строк, создатель должен действовать либо по согласно варианту a), либо согласно варианту b), приведенным ниже, в зависимости от конкретной реализации (однако если возможно, следует выбирать первый вариант, поскольку он порождает наименьшее количество индексов, указывающих на одну и ту же строку символов), в противном случае (не существует идентичной строки символов) создатель должен действовать согласно варианту b). Варианты действий a) и b) приведены ниже:

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

b) выбрать вариант **literal-character-string**, назначить заданную строку символов **literal-character-string**, и либо:

   1) добавить идентичную строку символов в соответствующую таблицу строк и установить для компонента **add-to-table** значение **TRUE** (данный вариант b1 не должен использоваться в случае, если соответствующая таблица строк уже содержит 2^{20} записей); либо

      ПРИМЕЧАНИЕ 1. – Если соответствующая таблица строк уже содержит 2^{20} записей, возможен только вариант b2.

   2) установить для компонента **add-to-table** значение **FALSE**.

      ПРИМЕЧАНИЕ 2. – Результатом выбора варианта b1 будет появление более чем одной идентичной строки символов в текущем содержимом. Это не сказывается на последующей обработке строк символов (см. п. 7.14.8).

7.14.8 При обработке абстрактного значения этого типа ASN.1, представляющего строку символов заданной категории, обработчик должен определить строку символов, представляемую данным абстрактным значением, следующим образом:

a) Если выбран вариант **string-index**, то строка символов, представляемая данным абстрактным значением, должна являться строкой символов в соответствующей таблице строк, индекс словарной таблицы для которой равен значению **string-index**.

   ПРИМЕЧАНИЕ 1. – Если значение **string-index** превышает текущий размер этой словарной таблицы, имеет место ошибка при обработке документа Быстрого информационного набора.

b) Если выбран вариант **literal-character-string** и для компонента **add-to-table** установлено значение **TRUE**, то строка символов, представляемая данным абстрактным значением, должна представлять собой значение компонента **literal-character-string**. Обработчик должен добавить идентичную строку символов в соответствующую таблицу строк (см. п. 7.14.12).

   ПРИМЕЧАНИЕ 2. – Если соответствующая таблица строк уже содержит 2^{20} строк, то имеет место ошибка при обработке документа Быстрого информационного набора.

c) Если выбран вариант **literal-character-string** и для компонента **add-to-table** установлено значение **FALSE**, то строка символов, представляемая данным абстрактным значением, должна представлять собой значение компонента **character-string**.

7.14.9 Если обработчик по какой-либо причине, включая ограничения, связанные с конкретной реализацией, не может добавить строку в словарную таблицу, в случае, когда такое добавление необходимо для выполнения действий согласно п. 7.14.8 b, он должен прекратить обработку документа Быстрого информационного набора и объявить об ошибке.

7.15 Тип NameSurrogate

7.15.1 Тип **NameSurrogate**:

```
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 п. 7.2.1.

7.15.3 Тип **NameSurrogate** содержит три положительных целых числа (первые два являются необязательными), формирующих заменитель имени (см. п. 8.5.2).

ПРИМЕЧАНИЕ. – Данный тип встречается только в компоненте **initial-vocabulary** типа **Document**.

7.15.4 Компоненты **prefix-string-index** (если таковой присутствует), **namespace-name-string-index** (если таковой присутствует), и **local-name-string-index** должны быть больше нуля и не превышать количества записей в таблицах PREFIX, NAMESPACE NAME, и LOCAL-NAME исходного словаря соответственно.

ПРИМЕЧАНИЕ. – Если обработчик документа Быстрого информационного набора принимает решение не проверять выполнение данного ограничения, в результате дальнейшей обработки могут возникнуть уязвимости, связанные с безопасностью.

7.15.5 Компонент **prefix-string-index** должен присутствовать только в том случае, если присутствует компонент **namespace-name-string-index**.

7.16 Тип QualifiedNameOrIndex

7.16.1 Тип **QualifiedNameOrIndex**:

```
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 п. 7.13 и п. 7.2.1 соответственно.

7.16.3 Тип **QualifiedNameOrIndex** представляет уточненное имя (см. п. 3.4.11).

7.16.4 Абстрактное значение этого типа ASN.1 содержит либо три компонента, соответствующие префиксу, имени пространства имен и локальному имени уточненного имени (заданной категории), либо индекс словарно таблицы заменителя имени заданной категории в словарной таблице для этой категории уточненных имен (см. п. 8.5.2), называемой соответствующей таблицей имен.

ПРИМЕЧАНИЕ. – Категория всегда определена в связанном тексте предыдущих пунктов, когда используется этот тип.

7.16.5 Компонент **name-surrogate-index**, если таковой присутствует, должен содержать индекс словарной таблицы заместителя имени в соответствующей 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 уточненное имя (заданной категории), создатель должен выполнять действия, определенные в последующих подпунктах.

7.16.7.1 Должно быть проверено выполнение следующих условий в порядке, приводимом ниже:

a) существует ли префикс уточненного имени в текущем содержимом таблицы PREFIX, или уточненное имя не имеет префикса;

b) существует ли имя пространства имен уточненного имени в текущем содержимом таблицы NAMESPACE NAME, или уточненное имя не имеет имени пространства имен;

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

d) первые три условия выполняются, и в текущем содержимом соответствующей таблицы имен существует заменитель имени (см. п. 8.5), состоящий из индекса (индексов) словарной таблицы для префикса (если таковой присутствует), имени пространства имен (если таковое присутствует) и локального имени.

7.16.7.2 Если все перечисленные выше условия выполнены, необходимо выбрать вариант использования **name-surrogate-index** и установить для этого компонента значение, равное индексу словарной таблицы заменителя имени, установленного в п. 7.16.7.1 d, в соответствующей таблице имен, завершая процедуры, предусмотренные п. 7.16.7.

7.16.7.3 В противном случае необходимо выбрать вариант использования **literal-qualified-name**, и все его компоненты должны быть назначены следующим образом:

a) если уточненное имя не имеет префикса, то компонент **prefix** не должен присутствовать, в противном случае префикс должен быть назначен компоненту **prefix** путем выполнения действий, указанных в п. 7.13.7 с тем ограничением, что перечень действий, указанный в п. 7.13.7 b) не должен выполняться, если идентичная строка символов существует в текущем содержимом соответствующей таблицы строк. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории PREFIX;

b) если уточненное имя не имеет имени пространства имен, то компонент **namespace-name** не должен присутствовать, в противном случае имя пространства имен должно быть назначено компоненту **namespace-name** путем выполнения действий, указанных в п. 7.13.7 с тем ограничением, что перечень действий, указанный в п. 7.13.7 b) не должен выполняться, если идентичная строка символов существует в текущем содержимом соответствующей таблицы строк. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории NAMESPACE NAME (см. п. 8.4.2);

c) локальное имя уточненного имени должно быть назначено (путем выполнения действий, указанных в п. 7.13.7) компоненту **local-name**. Данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории LOCAL NAME (см. п. 8.4.2).

ПРИМЕЧАНИЕ. – Выполнение действий, указанных в п. 7.13.7 в рамках данного подпункта может повлечь за собой добавление локального имени в таблицу LOCAL NAME.

7.16.7.4 Если выполнение действий, указанных в п. 7.13.7 в рамках подпункта 7.16.7.3 к одной или более из трех приведенных выше строк символов не повлекло за собой добавления строк в словарную таблицу в связи с тем, что в ней уже есть 2^{20} записей (см. п. 7.13.7 b), то заменитель имени для данного уточненного имени не может быть создан.

7.16.7.5 В противном случае должен быть создан заменитель имени (см. п. 8.5), состоящий из индекса (индексов) словарных таблиц для префикса (если таковой присутствует), имени пространства имен (если таковое присутствует) и локального имени. Если этот заменитель имени не существует в текущем содержимом соответствующей таблицы имен, то он должен быть добавлен в эту таблицу во всех случаях, кроме тех, когда она уже содержит 2^{20} записей.

7.16.8 При обработке абстрактного значения типа **QualifiedNameOrIndex**, представляющего уточненное имя заданной категории, обработчик должен определять уточненное имя, представленное данным абстрактным значением, в соответствии с последующими подпунктами.

7.16.8.1 Если выбран вариант использования **name-surrogate-index**, то уточненное имя, представленное данным абстрактным значением, должно быть представлено заменителем имени заданной категории (см. п. 8.5.2) в соответствующей таблице имен, индекс словарной таблицы которого равен значению **name-surrogate-index**.

7.16.8.2 Если выбран вариант использования **literal-qualified-name**, то:

a) Уточненное имя, представленное данным абстрактным значением, определяется следующим образом:

1) префикс уточненного имени должен быть определен (путем выполнения действий, указанных в п. 7.13.8) из компонента **prefix** (если таковой присутствует); данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории PREFIX;

2) имя пространства имен уточненного имени должно быть определено (путем выполнения действий, указанных в п. 7.13.8) из компонента **namespace-name** (если таковой присутствует); данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории NAMESPACE NAME (см. п. 8.4.2);

3) локальное имя уточненного имени должно быть определено (путем выполнения действий, указанных в п. 7.13.8) из компонента **local-name**; данный компонент относится к типу **IdentifyingStringOrIndex** (см. п. 7.13), представляющему здесь строку символов категории LOCAL NAME (см. п. 8.4.2);

ПРИМЕЧАНИЕ. – Эти действия могут потребовать добавления в соответствующие словарные таблицы, как определено в п. 7.13.8 b.

b) Если после возможных добавлений в результате обработки компонентов **literal-qualified-name** доступны индексы словарных таблиц для всех присутствующих компонентов, то заменитель имени, состоящий из этих индексов словарных таблиц, должен быть добавлен в соответствующую таблицу имен (но см. п. 7.16.9) во всех случаях, за исключением тех, когда данная таблица уже содержит 2^{20} заменителей имен.

7.16.9 Если обработчик по какой-либо причине, включая ограничения, связанные с конкретной реализацией, не может добавить заменитель имени в словарную таблицу, содержащую менее чем 2^{20} записей в случае, когда такое добавление необходимо для выполнения действий согласно п. 7.16.8.2 b, он должен прекратить обработку документа Быстрого информационного набора и объявить об ошибке.

7.17 Тип EncodedCharacterString

7.17.1 Тип **EncodedCharacterString**:

```
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, полученную в результате установления соответствия между строкой символов и строкой октетов, причем данное преобразование является обратимым. Создатель документа Быстрого информационного набора определяет в **encoding-format** способ кодирования, который будет использоваться, а обработчик использует эту информацию для декодирования октетов в компоненте **octets** и преобразования их обратно в строку символов.

7.17.3 Компонент **octets** должен нести значение, представляющее собой строку октетов, являющуюся кодированием строки символов способом, определенным в компоненте **encoding-format**.

ПРИМЕЧАНИЕ. – В целом существует несколько способов кодирования, которые могут быть использованы для кодирования заданной строки символов. Создатель документа Быстрого информационного набора может выбрать способ кодирования, основываясь на определенных критериях (например, он может постараться оптимизировать размер или скорость обработки), однако он также может предпочесть удобство и универсальную применимость UTF-8 или UTF-16BE. Также следует отметить, что некоторые способы кодирования способны закодировать только строки символов, которые содержать только подмножество символов ИСО/МЭК 10646.

7.17.4 Формат кодирования **`utf-8`** может быть использован для любой строки символов. Данный формат кодирования необходимо применять путем создания кодирования UTF-8 (см. ИСО/МЭК 10646) строки символов и назначения этого кодирования компоненту **`octets`**.

ПРИМЕЧАНИЕ. – Этот формат кодирования наиболее подходит для строк символов, в которых наиболее часто встречаются символы ИСО/МЭК 10646 из ранней части Основной многоязычной плоскости ИСО/МЭК 10646, и для которых никакие другие форматы кодирования не являются применимыми или более эффективными.

7.17.5 Формат кодирования **`utf-16`** может быть использован для любой строки символов. Данный формат кодирования необходимо применять путем создания кодирования UTF-16BE (см. Unicode, 2.6) строки символов и назначения этого кодирования компоненту **`octets`**.

ПРИМЕЧАНИЕ 1. – Этот формат кодирования наиболее подходит для строк символов, в которых присутствует широкий спектр символов ИСО/МЭК 10646, и для которых никакие другие форматы кодирования не являются применимыми или более эффективными.

ПРИМЕЧАНИЕ 2. – Порядок байтов кодирования UTF-16BE, определенный в Unicode, 2.6 – самый старший байт первый (более ранний байт в строке октетов).

7.17.6 Формат кодирования **`restricted-alphabet`** основывается на использовании ограниченного алфавита, выбираемого из числа присутствующих в таблице ограниченных алфавитов. Компонент **`restricted-alphabet`** должен содержать индекс словарной таблицы ограниченного алфавита. Этот формат кодирования может быть использован только в том случае, если строка символов полностью состоит из символов ограниченного алфавита, находящегося в записи таблицы ограниченных алфавитов, имеющей индекс равный значению компонента **`restricted-alphabet`**. Формат кодирования **`restricted-alphabet`** должен применяться, как определено в п. от 7.17.6.1 до п. 7.17.6.6.

7.17.6.1 Каждому символу ограниченного алфавита по порядку должно быть присвоено целочисленное значение (начиная с нуля).

7.17.6.2 Каждый символ в строке символов должен быть преобразован в целое число, которое назначено этому символу в данном ограниченном алфавите.

7.17.6.3 Каждое целое число должно быть представлено как целое двоичное число без знака в битовом поле. Размер битового поля должен определяться целочисленным значением, назначенным последнему символу в ограниченном алфавите. Это целочисленное значение должно быть увеличено на 1, чтобы получить целочисленное значение (скажем, N). Размер битового поля должен быть равен минимальному количеству битов, необходимых для того, чтобы закодировать N как кодирование целого числа без знака.

ПРИМЕЧАНИЕ 1. – Увеличение необходимо, поскольку значение, полностью состоящее из единиц в битовом поле, интерпретируется как конец строки символов и не может быть использовано для представления символа. Это означает, что если в ограниченном алфавите содержится некоторое число символов, представляющих собой число 2, возведенное в некоторую степень, битовое поле будет содержать на один бит больше, чем могло бы, не будь в составе ограниченного алфавита упомянутых символов.

ПРИМЕЧАНИЕ 2. – Например, если в ограниченном алфавите содержится 24 символа, то каждый из символов будет кодироваться в 5 битов, но если в ограниченном алфавите 32 символа, то каждый из них будет кодироваться в 5 битов.

7.17.6.4 Все эти битовые поля должны быть соединены (по порядку) в битовую строку.

7.17.6.5 Если длина полученной битовой строки не равна произведению целого числа на 8 битов, то биты "1" должны быть добавлены в конец строки так, чтобы ее длина стала равна произведению целого числа на 8 битов.

7.17.6.6 Полученная битовая строка (являющаяся теперь произведением целого числа на 8 битов), представленная как строка октетов, должна быть назначена компоненту **`octets`**.

7.17.7 Формат кодирования **`encoding-algorithm`** определяется алгоритмом кодирования (см. п. 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 назначен компоненту **`encoding-algorithm`** , и полученная в результате кодирования строка октетов должна быть назначена компоненту **`octets`**.

8 Построение и обработка документа Быстрого информационного набора

В документе Быстрого информационного набора широко используются индексы словарных таблиц разнообразных таблиц, создаваемых на различных этапах построения и обработки этого документа. В подпункте 8.1 определяется концептуальный порядок компонентов абстрактного значения типа **`Document`**, для того, чтобы удостовериться, что создатель и обработчики документа Быстрого информационного набора создают идентичные словарные таблицы. В последующих подпунктах определяются словарные таблицы, которые создаются и используются при создании и обработке документа Быстрого информационного набора. Представление этих таблиц в памяти компьютера зависит от конкретной реализации и не стандартизировано. Словарная таблица обеспечивает установление соответствия между индексом словарной таблицы и информацией в Информационном наборе XML (возможно, косвенно).

ПРИМЕЧАНИЕ. – Словарные таблицы для документа Быстрого информационного набора создаются динамически в процессе построения документа Быстрого информационного набора. Они создаются динамически из содержимого документа Быстрого информационного набора в процессе обработки документа Быстрого информационного набора. Обмен ими в какой-либо форме не имеет места.

8.1 Концептуальный порядок компонентов абстрактного значения типа `Document`

8.1.1 Для обеспечения единообразия способа назначения индексов словарных таблиц при создании и обработке документа Быстрого информационного набора в рамках различных реализаций, для компонентов абстрактного значения типа **`Document`** определен концептуальный порядок. При создании и обработке таких абстрактных значений должен использоваться этот концептуальный порядок при добавлении строк (см. п. 7.13.7 и п. 7.14.6) и заменителей имен (см. п. 7.16.7) в словарные таблицы.

ПРИМЕЧАНИЕ. – Этот порядок совпадает с порядком кодирований компонентов в документе Быстрого информационного набора. Это не обязательно подразумевает, что значение, которое несет документ, обрабатывается в этом порядке. Этот порядок определен единственно для того, чтобы удостовериться, что один и тот же индекс словарной таблицы будет назначен данной записи словарной таблицы как создателем, так и обработчиком документа Быстрого информационного набора.

8.1.2 Концептуальный порядок для создания и обработки документа Быстрого информационного набора определяется следующим образом: Компоненты абстрактного значения типа **`Document`** должны быть рассмотрены в соответствии с алгоритмом, определенным в п. от 8.1.2.1 до п. 8.1.2.5. Порядок рассматривания компонентов определяет концептуальный порядок.

8.1.2.1 Компонент абстрактного значения (соответствующего типу **`Document`**) должен быть рассмотрен первым.

8.1.2.2 Если рассматриваемый компонент относится к типу sequence, то компоненты типа sequence, которые присутствуют в данном абстрактном значении, должны быть рассмотрены в порядке их определения по тексту, от первого присутствующего компонента к последнему, с выполнением действий, указанных в п. от 8.1.2.1 до п. 8.1.2.5, рекурсивно над каждым рассматриваемым компонентом.

8.1.2.3 Если рассматриваемый компонент относится к типу sequence-of, то все случаи появления компонента sequence-of должны быть рассмотрены в порядке sequence-of, от первого случая появления компонента sequence-of к последнему, с выполнением действий, указанных в п. от 8.1.2.1 до п. 8.1.2.5, рекурсивно над каждым рассматриваемым компонентом.

8.1.2.4 Если рассматриваемый компонент относится к типу choice, то вариант, присутствующий в абстрактном значении должен быть рассмотрен и действия, указанные в п. от 8.1.2.1 до п. 8.1.2.5, должны быть выполнены рекурсивно над данным вариантом.

8.1.2.5 Если рассматриваемый компонент относится к другому типу ASN.1, то никаких дальнейших действий над данным компонентом выполнять не требуется.

8.2 Таблица ограниченных алфавитов

8.2.1 С каждым документом Быстрого информационного набора связана таблица ограниченных алфавитов. Таблица ограниченных алфавитов содержит ограниченные алфавиты, к которым можно обращаться посредством индексов словарных таблиц.

8.2.2 Каждая запись в таблице ограниченных алфавитов должна представлять собой упорядоченный набор неодинаковых символов ИСО/МЭК 10646 размером от 2 до 2^{20} символов.

ПРИМЕЧАНИЕ. – Ограниченный алфавит позволяет осуществлять компактное кодирование любой строки символов, целиком состоящей из символов этого набора, путем назначения символам в наборе увеличивающихся целочисленных значений и использования этих целых чисел для кодирования символов строки (см. п. 7.17.6).

8.3 Таблица алгоритмов кодирования

8.3.1 С каждым документом Быстрого информационного набора связана таблица алгоритмов кодирования. Таблица алгоритмов кодирования содержит определения алгоритмов кодирования, к которым можно обращаться посредством индексов словарных таблиц.

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

ПРИМЕЧАНИЕ. – Заданные характеристики могут касаться длины строки, символов, встречающихся в ней, или образца последовательности символов произвольной сложности. В целом, данный конкретный алгоритм кодирования применяется только к специальному и определенному подмножеству строк символов ИСО/МЭК 10646.

8.3.3 На алгоритмы кодирования накладываются следующие ограничения:

a) алгоритм кодирования должен иметь связанный с ним URI (кроме случаев, когда алгоритм кодирования является встроенным) для того, чтобы на него можно было сослаться для добавления в таблицу ;

b) алгоритм кодирования должен точно определять, к каким видам строк символов он может быть применен; это определение должно включать ограниченный алфавит (если таковой присутствует), диапазон длин (если таковой присутствует) и любые дополнительные ограничения на длину и содержание строк символов (например, образец);

c) для любой строки символов, к которой он может быть применен, алгоритм кодирования должен предоставлять установление соответствия между этой строкой символов и строкой октетов, причем этот процесс должен быть осуществим и в обратном порядке.

ПРИМЕЧАНИЕ 1. – Описанное выше подразумевает, что не может существовать строка символов S, для которой кодирование S – E, за которым следует декодирование S – S', так, что $S' \neq S$, даже если различие между S' и S незначительно (например, дополнительный пробел). С другой стороны, не требуется, чтобы имелась возможность закодировать любую строка символов S, а также не требуется, чтобы кодировки были стандартными.

ПРИМЕЧАНИЕ 2. – От приложения, создающего документ Быстрого информационного набора на основе данных, находящихся в памяти, таких, как числа с плавающей точкой, не требуется вначале 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, что могла бы быть получена путем применения этого алгоритма кодирования к строке символов, которую представляют данные в памяти, и что к данной строке символов может быть применен этот алгоритм кодирования.

ПРИМЕЧАНИЕ 3. – Алгоритмы кодирования (за исключением встроенных) могут быть определены в других стандартах или быть взаимно согласованы между создателем и обработчиком документа Быстрого информационного набора.

8.4 Динамические таблицы строк

8.4.1 С каждым документом Быстрого информационного набора связаны восемь динамических таблиц строк. Каждая из динамических таблиц строк содержит строки символов, к которым можно обращаться посредством индексов словарных таблиц.

8.4.2 В настоящей Рекомендации | Международном стандарте все строки символов, встречающиеся в документе Быстрого информационного набора, классифицируются по восьми приведенным ниже категориям, каждая из которых имеет динамическую таблицу строк:

a) PREFIX: Данная категория заключает в себе строки символов, являющиеся свойством **[prefix]** единиц информации **element**, **attribute**, или **namespace**.

b) NAMESPACE NAME: Данная категория заключает в себе строки символов, являющиеся свойством **[namespace name]** единиц информации **element, attribute**, или **namespace**.

c) LOCAL NAME: Данная категория заключает в себе строки символов, являющиеся свойством **[local name]** единиц информации **element** или **attribute**.

d) OTHER NCNAME: Данная категория заключает в себе строки символов, являющиеся свойством **[target]** единиц информации **processing instruction**; или свойством **[name]** единиц информации **unexpanded entity reference**, **unparsed entity**, или **notation**; или свойством **[notation name]** единицы информации **unparsed entity**.

e) OTHER URI: Данная категория заключает в себе строки символов, являющиеся свойством **[system identifier]** свойства **[public identifier]** единиц информации **unexpanded entity reference**, **document type declaration**, **unparsed entity**, или **notation**.

f) ATTRIBUTE VALUE: Данная категория заключает в себе строки символов, являющиеся свойством **[normalized value]** единицы информации **attribute**.

g) CONTENT CHARACTER CHUNK: Данная категория заключает в себе строки символов, являющиеся свойством **[character code]** блока единиц информации **character,** являющихся последовательными дочерними элементами данной единицы информации **element**.

h) OTHER STRING: Данная категория заключает в себе строки символов, являющиеся свойством **[version]** единицы информации **document**; или свойством **[content]** единиц информации **processing instruction** или **comment**.

8.5 Динамические таблицы имен и заменители имен

8.5.1 С каждым документом Быстрого информационного набора связаны две динамических таблиц имен. Каждая из динамических таблиц имен содержит заменители имен, к которым можно обращаться посредством индексов словарных таблиц и которые используются для идентификации уточненного имени, которое может либо иметь префикс, либо не иметь его (а также может иметь или не иметь имя пространства имен).

8.5.2 Заменитель имени представляет собой набор из, как максимум, трех индексов словарных таблиц:

a) (необязательный) индекс строки в таблице PREFIX;

b) (необязательный) индекс строки в таблице NAMESPACE NAME; и

c) индекс 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, и заменитель имени представляет уточненное имя с префиксом;

b) присутствуют только второй и третий индексы, и заменитель имени представляет уточненное имя без префикса, которое имеет имя пространства имен;

c) присутствует третий индекс, и заменитель имени представляет уточненное имя без префикса, которое не имеет имени пространства имен.

8.5.4 В настоящей Рекомендации | Международном стандарте все уточненные имена, встречающиеся в документе Быстрого информационного набора, классифицируются по двум приведенным ниже категориям, каждая из которых имеет динамическую таблицу имен:

a) ELEMENT NAME: Данная категория заключает в себе заменители имен, представляющие уточненное имя единицы информации **element**.

b) ATTRIBUTE NAME: Данная категория заключает в себе заменители имен, представляющие уточненное имя единицы информации **attribute**.

8.5.5 Уточненное имя, представленное заданным заменителем имени, должно определяться следующим образом (при условии, что дана динамическая таблица строк):

a) первый индекс словарной таблицы (если таковой присутствует) должен интерпретироваться как индекс словарной таблицы строки символов в таблице PREFIX; и

b) второй индекс словарной таблицы (если таковой присутствует) должен интерпретироваться как индекс словарной таблицы строки символов в таблице NAMESPACE NAME; и

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

9 Встроенные ограниченные алфавиты

9.1 "Цифровой" ограниченный алфавит

9.1.1 Данному ограниченному алфавиту присвоен индекс словарной таблицы 1. Этот алфавит состоит из следующих пятнадцати символов ИСО/МЭК 10646 (в порядке, приведенном ниже):

ЦИФРА от НУЛЯ до ДЕВЯТИ (DIGIT ZERO to DIGIT NINE)

ПЕРЕНОС-МИНУС (HYPHEN-MINUS)

ЗНАК ПЛЮС (PLUS SIGN)

ТОЧКА (FULL STOP)

ЛАТИНСКАЯ СТРОЧНАЯ БУКВА E (LATIN SMALL LETTER E)

ПРОБЕЛ (SPACE)

9.1.2 Данный ограниченный алфавит подходит для кодирования строк символов, представляющих несколько типов чисел, в том числе числа с плавающей точкой в научной нотации. Отдельная строка символов может содержать несколько чисел, разделенных пробелами.

9.2 Ограниченный алфавит "дата и время"

9.2.1 Данному ограниченному алфавиту присвоен индекс словарной таблицы 2. Этот алфавит состоит из следующих пятнадцати символов ИСО/МЭК 10646 (в порядке, приведенном ниже):

ЦИФРА от НУЛЯ до ДЕВЯТИ (DIGIT ZERO to DIGIT NINE)

ПЕРЕНОС-МИНУС (HYPHEN-MINUS)

ДВОЕТОЧИЕ (COLON)

ЛАТИНСКАЯ ЗАГЛАВНАЯ БУКВА T (LATIN CAPITAL LETTER T)

ЛАТИНСКАЯ ЗАГЛАВНАЯ БУКВА Z (LATIN CAPITAL LETTER Z)

ПРОБЕЛ (SPACE)

9.2.2 Данный ограниченный алфавит подходит для кодирования строк символов, представляющих наиболее часто встречающиеся выражения, обозначающие дату и время согласно ISO 8601. Отдельная строка может содержать несколько таких выражений, разделенных пробелами.

10 Встроенные алгоритмы кодирования

10.1 Общие положения

10.1.1 В данном пункте определяются встроенные алгоритмы кодирования. Встроенные алгоритмы кодирования не имеют связанного с ними URI.

ПРИМЕЧАНИЕ. – URI необходимы для алгоритмов кодирования, которые должны быть явным образом идентифицированы в исходном словаре, но встроенные алгоритмы кодирования всегда добавляются в таблицу алгоритмов кодирования неявным образом, и в связи с этим им не требуется URI.

10.1.2 В рамках данного пункта термин "слово" указывает на любую группу последовательных символов внутри заданной строки символов, которая:

a) не содержит ПРОБЕЛОВ;

b) либо находится в начале строки символов, либо предваряется ПРОБЕЛОМ;

c) либо находится в конце строки символов, либо за ним следует ПРОБЕЛ.

ПРИМЕЧАНИЕ. – "Слово" не обязательно состоит только из алфавитных (буквенных) символов.

10.2 "Шестнадцатеричный" алгоритм кодирования

10.2.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 1. Он может быть применен только к строкам символов, состоящим из следующих шестнадцати символов ИСО/МЭК 10646:

ЦИФРА от НУЛЯ до ДЕВЯТИ

ЛАТИНСКАЯ ЗАГЛАВНАЯ БУКВА от A до F

и содержащим четное количество символов (включая ноль).

ПРИМЕЧАНИЕ. – Использование встроенных пробельных символов XML не разрешено.

10.2.2 Строка символов должна интерпретироваться как шестнадцатеричное кодирование строки октетов, где первый символ строки соответствует самому старшему полубайту первого октета, и т.д.

10.3 Алгоритм кодирования "base64"

10.3.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 2. Он может быть применен только к строкам символов, которые:

a) состоят полностью из символов ЛАТИНСКАЯ ЗАГЛАВНАЯ БУКВА от A до Z, ЛАТИНСКАЯ СТРОЧНАЯ БУКВА от A до Z, ЦИФРА от НУЛЯ до ДЕВЯТИ, ЗНАК ПЛЮС, ДРОБНАЯ ЧЕРТА и ЗНАК РАВЕНСТВА; и

ПРИМЕЧАНИЕ. – Присутствие в строке символов пробельных символов XML не допускается.

b) либо представляет собой корректный экземпляр Кодирования передаваемого содержания, определенной в IETF RFC 2045, 6.8, либо становится корректным экземпляром этого кодирования посредством вставки пробельных символов XML там, где это требуется согласно IETF RFC 2045.

10.3.2 Строка символов должна интерпретироваться как кодирование Base64 (см. IETF RFC 2045) строки октетов (подразумевается, что пробельные символы XML присутствуют там, где это требуется согласно IETF RFC 2045). Получаемая в результате строка октетов является строкой октетов, определенной этим кодированием Base64.

10.3.3 Данный алгоритм кодирования подходит для кодирования строк символов, которые не содержат пробельных символов XML и которые являются строками Base64 (любой длины) или станут или после добавления пробельных символов XML.

10.4 Алгоритм кодирования "короткое целое"

10.4.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 3. Он может быть применен только к строкам символов, которые удовлетворяют следующим требованиям:

a) строка символов состоит полностью из символов ЦИФРА от НУЛЯ до ДЕВЯТИ, ПЕРЕНОС-МИНУС и ПРОБЕЛ; и

b) ни первый, ни последний символы не являются символами ПРОБЕЛ, и в строке нет двух соседних символов ПРОБЕЛ; и

c) строка символов содержит как минимум одно слово (см. п. 10.1.2); и

d) все присутствующие символы ПЕРЕНОС-МИНУС являются первыми символами слов; и

e) за каждым из символов ПЕРЕНОС-МИНУС следует как минимум один символ ЦИФРА от ЕДИНИЦЫ до ДЕВЯТИ; и

f) каждый символ ЦИФРА НОЛЬ либо является единственным символом в слове, либо предваряется символом ЦИФРА от ЕДИНИЦЫ до ДЕВЯТИ; и

g) каждое слово в строке символов, будучи интерпретировано как строка цифровых символов – целое десятичное число со знаком, возвращает значение в диапазоне от –32768 до 32767.

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

10.4.3 Каждая группа из 8 битов в 16-битном целом с поразрядным дополнением до двух для слова должна производить октет строки октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов должен становиться самым старшим битом соответствующего октета. Если в строке символов несколько слов, они должны кодироваться по порядку, и октеты, производимые множественными 16-битными целыми с поразрядным дополнением до двух должны соединяться в этом порядке.

10.4.4 Данный алгоритм кодирования подходит для кодирования строк символов, представляющих единичные целые числа в диапазоне от –32768 до 32767 (которые можно представить как 16-битное целое с поразрядным дополнением до двух) или списки таких чисел.

10.5 Алгоритм кодирования "целое"

10.5.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 4. Он может быть применен только к строкам символов, которые удовлетворяют следующим требованиям:

a) строка символов удовлетворяет требованиям, определенным в п. 10.4.1 a - f; и

b) каждое слово в строке символов, будучи интерпретировано как строка цифровых символов – целое десятичное число со знаком, возвращает значение в диапазоне от –2147483648 до 2147483647.

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

10.5.3 Каждая группа из 8 битов в 32-битном целом с поразрядным дополнением до двух для слова должна производить октет строки октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов должен становиться самым старшим битом соответствующего октета. Если в строке символов несколько слов, они должны кодироваться по порядку, и октеты, производимые множественными 32-битными целыми с поразрядным дополнением до двух должны соединяться в этом порядке.

10.5.4 Данный алгоритм кодирования подходит для кодирования строк символов, представляющих единичные целые числа в диапазоне от –2147483648 до 2147483647 (которые можно представить как 32-битное целое с поразрядным дополнением до двух) или списков таких чисел.

10.6 Алгоритм кодирования "целое"

10.6.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 5. Он может быть применен только к строкам символов, которые удовлетворяют следующим требованиям:

a) строка символов удовлетворяет требованиям, определенным в п. 10.4.1 a - f; и

b) каждое слово в строке символов, будучи интерпретировано как строка цифровых символов – целое десятичное число со знаком, возвращает значение в диапазоне от –9223372036854775808 до 9223372036854775807.

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

10.6.3 Каждая группа из 8 битов в 64-битном целом с поразрядным дополнением до двух для слова должна производить октет строки октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов должен становиться самым старшим битом соответствующего октета. Если в строке символов несколько слов, они должны кодироваться по порядку, и октеты, производимые множественными 64-битными целыми с поразрядным дополнением до двух должны соединяться в этом порядке.

10.6.4 Данный алгоритм кодирования подходит для кодирования строк символов, представляющих единичные целые числа в диапазоне от –9223372036854775808 до 9223372036854775807 (которые можно представить как 64-битное целое с поразрядным дополнением до двух) или списки таких чисел.

10.7 "Булев" алгоритм кодирования

10.7.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 6. Он может быть применен только к строкам символов, которые удовлетворяют следующим требованиям:

a) строка символов состоит полностью из одного или нескольких слов "истина" ("true") и "ложь" ("false") и символов ПРОБЕЛ; и

b) ни первый, ни последний символы с строке символов не являются символами ПРОБЕЛ, и в строке символов нет двух соседних символов ПРОБЕЛ; и

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

10.7.2 Каждое слово "ложь" или "истина" в строке символов должно быть закодировано как один бит (для которого должно быть установлено либо значение ноль, либо единица соответственно) производимой строки октетов, начиная с пятого бита первого октета до восьмого бита первого октета. Последующие биты помещаются в последующие октеты, начиная с первого бита каждого октета до восьмого бита этого октета, с использованием такого количества октетов, которое требуется. Для любых неиспользуемых битов последнего октета должно быть установлено значение ноль.

10.7.3 В первых четырех битах первого октета должно содержаться количество неиспользуемых битов в последнем октете, закодированное как 4-битное целое число без знака.

ПРИМЕЧАНИЕ. – Первый октет может быть также и последним и содержать, как максимум, три неиспользованных бита. Если присутствует более чем один октет, то последний октет может содержать как максимум семь неиспользованных битов.

10.8 Алгоритм кодирования "число с плавающей точкой"

10.8.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 7. Он может быть применен только к строкам символов, которые удовлетворяют следующим требованиям:

a) строка символов состоит полностью из символов ЦИФРА от НУЛЯ до ДЕВЯТИ, ПЕРЕНОС-МИНУС, ТОЧКА, ЛАТИНСКАЯ ЗАГЛАВНАЯ БУКВА E и ПРОБЕЛ; и

ПРИМЕЧАНИЕ. – Использование ЛАТИНСКОЙ СТРОЧНОЙ БУКВЫ e не допускается, поскольку в этом случае не будет возможности осуществить обратное преобразование.

b) ни первый, ни последний символы с строке символов не являются символами ПРОБЕЛ, и в строке символов нет двух соседних символов ПРОБЕЛ; и

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

d) каждое слово в строке символов соответствует каноническому лексическому представлению числа с плавающей точкой, как определено в Схеме XML W3C, Часть 2, 3.2.4; и

e) каждое слово (см. п. 10.1.2) в строке символов, будучи интерпретировано как строка цифровых символов – десятичных чисел с плавающей точкой, возвращает значение, которое возможно представить как 32-битное число с плавающей точкой IEEE 754.

10.8.2 Каждое слово (см. п. 10.1.2) в строке символов должно быть интерпретировано как строка цифровых символов – десятичных чисел с плавающей точкой, и должно быть представлено как 32-битное число с плавающей точкой IEEE 754.

10.8.3 Каждая группа из 8 битов в 32-битном числе с плавающей точкой IEEE 754 для слова должна производить октет строки октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов должен становиться самым старшим битом соответствующего октета. Если в строке символов несколько слов, они должны кодироваться по порядку, и октеты, производимые множественными 32-битными числами с плавающей точкой IEEE 754 должны соединяться в этом порядке.

10.8.4 Данный алгоритм кодирования подходит для кодирования строк символов, представляющих единичные числа, представляемые как 32-битные числа с плавающей точкой IEEE 754 или списков таких чисел с плавающей точкой.

10.9 Алгоритм кодирования "число с плавающей точкой двойной точности"

10.9.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 8. Он может быть применен только к строкам символов, которые удовлетворяют следующим требованиям:

a) строка символов удовлетворяет требованиям, определенным в п. 10.8.1 a - c;

b) каждое слово в строке символов соответствует каноническому лексическому представлению числа с плавающей точкой, как определено в Схеме XML W3C, Часть 2, 3.2.5; и

c) каждое слово (см. п. 10.1.2) в строке символов, будучи интерпретировано как строка цифровых символов – десятичных чисел с плавающей точкой, возвращает значение, которое возможно представить как 64-битное число с плавающей точкой двойной точности IEEE 754.

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

10.9.3 Каждая группа из 8 битов в 64-битном числе с плавающей точкой двойной точности IEEE 754 для слова должна производить октет строки октетов, начиная с самой старшей группы из 8 битов. Самый старший бит в каждой группе из 8 битов должен становиться самым старшим битом соответствующего октета. Если в строке символов несколько слов, они должны кодироваться по порядку, и октеты, производимые множественными 64-битными числами с плавающей точкой двойной точности IEEE 754 должны соединяться в этом порядке.

10.9.4 Данный алгоритм кодирования подходит для кодирования строк символов, представляющих единичные числа, представляемые как 64-битные числа с плавающей точкой двойной точности IEEE 754 или списков таких чисел с плавающей точкой.

10.10 Алгоритм кодирования "uuid"

10.10.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 8. Он может быть применен только к строкам символов, которые удовлетворяют следующим требованиям:

a) строка символов состоит полностью из символов ЦИФРА от НУЛЯ до ДЕВЯТИ, ЛАТИНСКАЯ СТРОЧНАЯ БУКВА от A до F, ПЕРЕНОС-МИНУС и ПРОБЕЛ; и

b) ни первый, ни последний символы с строке символов не являются символами ПРОБЕЛ, и в строке символов нет двух соседних символов ПРОБЕЛ; и

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

d) каждое слово содержит ровно 36 символов; и

e) в каждом слове содержится ровно четыре символа ПЕРЕНОС-МИНУС, занимающих позиции 9, 14, 19 и 24 (считая от единицы).

10.10.2 Каждое слово в строке символов должно интерпретироваться как шестнадцатеричное представление UUID (см. Рек. МСЭ-Т X.667 | ИСО/МЭК 9834-8, 6.4), и должно быть представлено как 16-битное целое число без знака, как определено в Рек. МСЭ-Т X.667 | ИСО/МЭК 9834-8, 6.3. Если присутствует несколько слов, то несколько 16-октетных целых чисел без знака должны быть соединены.

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

10.11 Алгоритм кодирования "cdata"

10.11.1 Данному алгоритму кодирования присвоен индекс словарной таблицы 8. Он может быть применен только к любой строке символов.

10.11.2 Производимая строка октетов должна представлять собой кодирование UTF-8 (см. ИСО/МЭК 10646) строки символов.

10.11.3 Данный алгоритм должен использоваться только с Информационными наборами XML, созданными путем анализа документа XML и там, где дополнительная информация указывает, что строка символов соответствует целой секции CDATA (см. XML W3C 1.0 и XML W3C 1.1). Если данный алгоритм кодирования используется внутри документа Быстрого информационного набора, то ко всем строкам символов, соответствующим секциям CDATA, должны быть применен этот алгоритм кодирования.

11 Ограничения на поддерживаемые Информационные наборы XML и прочие упрощения

11.1 В рамках настоящей Рекомендации | Международного стандарта поддерживается большинство Информационных наборов XML, которые встречаются на практике, однако не поддерживаются некоторые Информационные наборы XML, которые возможны теоретически, но в нормальных обстоятельствах не возникают.

11.2 Термин "XML-самосогласованный" используется в настоящем пункте в следующем значении: набор свойств одной или более единиц информации является "XML-самосогласованным", если этот набор свойств может быть получен путем анализа подходящего корректного с точки зрения пространств имен документа XML.

11.3 Поддерживаемые Информационные наборы XML отвечают следующим условиям:

a) для свойства **[all declarations processed]** единицы информации **document** установлено значение **true**;

b) свойство **[in-scope namespaces]** каждой единицы информации **element** вместе со свойством **[namespace attributes]** всех единиц информации **element** формируют XML-самосогласованный набор;

c) свойство **[namespace name]** каждой единицы информации **element** вместе со свойством **[namespace attributes]** всех единиц информации **element** и свойством **[prefix]** единицы информации **element** формируют XML-самосогласованный набор;

d) свойство **[namespace name]** каждой единицы информации **attribute** вместе со свойством **[namespace attributes]** всех единиц информации **element** и свойством **[prefix]** этой единицы информации **attribute** формируют XML-самосогласованный набор;

e) свойство **[references]** каждой единицы информации **attribute** вместе со свойством **[normalized value]** единицы информации **attribute** формируют XML-самосогласованный набор;

f) свойство **[notation]** каждой единицы информации **processing instruction** вместе со свойством **[target]** единицы информации **processing instruction** вместе со свойством **[notations]** единицы информации **document** формируют XML-самосогласованный набор;

g) свойство **[notation]** каждой единицы информации **unparsed entity** вместе со свойством **[notation name]** единицы информации **unparsed entity** и свойством **[notations]** единицы информации **document** формируют XML-самосогласованный набор;

h) свойство **[element content whitespace]** всех единиц информации **character**, которые не представляют пробельные символы, имеет значение **false**;

i) свойство **[element content whitespace]** каждой единицы информации **character** вместе со свойством **[character code]** единицы информации **character** формируют XML-самосогласованный набор;

j) свойство **[normalized value]** всех единиц информации **attribute** и свойство **[content]** всех единиц информации **comment** и **processing instruction** содержат как максимум 2^{32} символов.

11.4 Приведенные ниже свойства единиц информации Информационного набора XML не включены в типы ASN.1, представляющие эти единицы информации:

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

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

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

d) свойства **[notation]** и **[parent]** единицы информации **processing instruction** (см. п. п. 7.5.7 и п. 7.5.8);

e) свойства **[declaration base URI]** и **[parent]** единицы информации **unexpиed entity reference** (см. п. 7.6.7 и п. 7.6.8);

f) свойство **[element content whitespace]** единицы информации **character** (см. п. 7.7.7);

g) свойство **[parent]** единицы информации **character** (см. п. 7.7.8);

h) свойство **[parent]** единицы информации **comment** (см. п. 7.8.6);

i) свойство **[parent]** единицы информации **document type declaration** (см. п. 7.9.7);

j) свойства **[declaration base URI]** и **[notation]** единицы информации **unparsed entity** (см. п. 7.10.8 и 7.10.9);

k) свойство **[declaration base URI]** единицы информации **notation** (см. п. 7.11.7).

12 Кодирование битового уровня типа `Document`

12.1 В данном пункте определяются специальные кодирования типа **`Document`** для формирования документа Быстрого информационного набора.

ПРИМЕЧАНИЕ. – Эти специальные кодирования разработаны для оптимизации скорости обработки и компактности, которые признаются наиболее важными во многих предполагаемых случаях использования настоящей Рекомендации | Международного стандарта.

12.2 Кодирования определены в терминах действий, в результате выполнения которых кодировщиком биты присоединяются к потоку битов. Исходный поток битов либо пуст, либо состоит из декларации XML (см. п. 12.3).

12.3 Декларация XML (см. XML W3C 1.1, 2.8) может (по усмотрению создателя) включаться в начало потока битов. Декларация XML (если таковая присутствует) должна представлять собой одну из следующих строк символов, закодированную 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 Для номера версии (если таковой присутствует) в декларации XML должно быть установлено значение соответствующего свойства **[version]** единицы информации **document**. Декларация XML не должна содержать номера версии в случае, если свойство **[version]** не имеет значения.

12.5 Для декларации автономности (если таковая присутствует) в декларации XML должно быть установлено значение соответствующего свойства **[standalone]** единицы информации **document**. Декларация XML не должна содержать декларации автономности в случае, если свойство **[standalone]** не имеет значения.

12.6 Шестнадцать битов '1110000000000000' должны быть присоединены к потоку битов.

ПРИМЕЧАНИЕ. – Эти биты будут находиться либо в начале документа Быстрого информационного набора, либо следовать за декларацией XML. В случае отсутствия декларации XML анализатор, просматривая первые 16 битов кодирования, может отличить потенциальный документ Быстрого информационного набора от любого другого корректного документа XML 1.0 W3C или XML 1.1 W3C, поскольку такие 16 битов никогда не встречаются в начале корректного документа XML.

12.7 Далее к потоку битов должно быть присоединено битовое поле из 16 битов, содержащее номер версии настоящей Рекомендации | Международного стандарта (см. п. 12.9), закодированное как 16-битное целое число без знака.

12.8 Далее к потоку битов должен быть присоединен бит '0' (дополнение).

ПРИМЕЧАНИЕ. – Это делается для обеспечения выравнивания байтов в последующих частях кодирования.

12.9 Для данной редакции настоящей Рекомендации | Международного стандарта номер версии 1.

ПРИМЕЧАНИЕ. – Предполагается, что для последующих редакций настоящей Рекомендации | Международного стандарта номер версии будет увеличиваться в случае, если возможно возникновение проблем взаимодействия между новой редакцией и предыдущими редакциями.

12.10 Кодирование ECN (см. Рек. МСЭ-Т X.692 | ИСО/МЭК 8825-3) абстрактного значения типа **`Document`**, определенное Модулем связи кодирования в п. A.2, должно быть присоединено к потоку битов.

ПРИМЕЧАНИЕ. – В Приложении C приводится неформальное описание кодирований, определенных в п. A.2.

12.11 Если кодирование абстрактного значения типа **`Document`** не заканчивается на последнем бите октета, то четыре бита '0000' (дополнение) должны быть присоединены к потоку битов для завершения последнего октета.

12.12 По выполнении всех шагов, приведенных выше, содержимое потока битов представляет собой документ Быстрого информационного набора.

Приложение A

Модуль ASN.1 и модули ECN для документов Быстрого информационного набора

(Данное Приложение является неотъемлемой частью настоящей Рекомендации | Международного стандарта)

A.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, 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 – В 20-ю степень

four-gig INTEGER ::= 4294967296 – В 32-ю степень

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
                                -- OTHER URI category --,
      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 automatically imports:
      #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)

-- Используется при кодировании длины SEQUENCE OF (см. C.21)

#NonEmptySequenceOfLengthAlternatives1 ::= #ALTERNATIVES {
      small         #INT(1..128),
      large         #INT(129..1048576) }
-- Используется при кодировании длины NonEmptyOctetString (см. C.22)

#NonEmptyOctetStringLengthAlternatives2 ::= #ALTERNATIVES {
      small         #INT(1..64),
      medium        #INT(65..320),
      large         #INT(321..4294967296) }

-- Используется при кодировании длины NonEmptyOctetString (см. C.23)

#NonEmptyOctetStringLengthAlternatives5 ::= #ALTERNATIVES {
      small         #INT(1..8),
      medium        #INT(9..264),
      large         #INT(265..4294967296) }

-- Используется при кодировании длины NonEmptyOctetString (см. C.24)

#NonEmptyOctetStringLengthAlternatives7 ::= #ALTERNATIVES {
      small         #INT(1..2),
      medium        #INT(3..258),
      large         #INT(259..4294967296) }

-- Используется при кодировании положительного целого числа (см. C.25)

#PositiveIntegerAlternatives2 ::= #ALTERNATIVES {
      small          #INT(1..64),
      medium         #INT(65..8256),
      large          #INT(8257..1048576) }

-- Используется при кодировании положительного целого числа (см. C.27)

#PositiveIntegerAlternatives3 ::= #ALTERNATIVES {
      small          #INT(1..32),
      medium         #INT(33..2080),
      medium-large   #INT(2081..526368),
      large          #INT(526369..1048576) }

-- Используется при кодировании положительного целого числа (см. C.28)

#PositiveIntegerAlternatives4 ::= #ALTERNATIVES {
      small          #INT(1..16),
      medium         #INT(17..1040),
      medium-large   #INT(1041..263184),
      large          #INT(263185..1048576) }

-- Используется при кодировании неотрицательного целого числа (см. C.26)

#NonNegativeIntegerAlternatives2 ::= #ALTERNATIVES {
      zero           #INT(0),
      small          #INT(1..64),
      medium         #INT(65..8256),
      large          #INT(8257..1048576) }

-- Используется для вставки начального дополнения во многих случаях

#PrecededByPrepadding{<#C>} ::= #CONCATENATION {
      prepadding       #PAD,
      original         #C }

-- Используется для вставки двухвариантного дискриминанта перед кодированием

#PrecededByTwoAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
      discriminant     #TwoAlternativeDiscriminant,
      original         #C }
```

```
-- Используется для вставки трехвариантного дискриминанта перед кодированием
#PrecededByThreeAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
      discriminant    #ThreeAlternativeDiscriminant,
      original        #C }

-- Используется для вставки четырехвариантного дискриминанта перед кодированием
#PrecededByFourAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
      prepadding      #PAD,
      discriminant    #FourAlternativeDiscriminant,
      original        #C }

-- Используется для вставки пятивариантного дискриминанта перед кодированием
#PrecededByFiveAlternativeDiscriminant{<#C>} ::= #CONCATENATION {
      prepadding      #PAD,
      discriminant    #FiveAlternativeDiscriminant,
      original        #C }

-- Используется для вставки поля длины перед кодированием SEQUENCE OF
#PrecededByNonEmptySequenceOfLength{<#C>} ::= #CONCATENATION {
      length          #NonEmptySequenceOfLength,
      original        #C }

-- Используется для вставки поля длины перед кодированием NonEmptyOctetString
#PrecededByNonEmptyOctetStringLength{<#C>} ::= #CONCATENATION {
      length          #NonEmptyOctetStringLength,
      original        #C }

-- Кодирует единицу компонента notations типа Document
-- (см. C.2.6.1)
eNotationDriver1 #Notation ::= {
      ENCODE STRUCTURE {
          STRUCTURED WITH eNotationPrepaddingAdder1 }
      WITH FastInfosetEncodingSet }

-- Кодирует единицу компонента unparsed-entities типа Document
--(см. C.2.7.1)
eUnparsedEntityDriver1 #UnparsedEntity ::= {
      ENCODE STRUCTURE {
          STRUCTURED WITH eUnparsedEntityPrepaddingAdder1 }
      WITH FastInfosetEncodingSet }

-- Кодирует единицу компонента namespace-attributes типа Element
--(см. C.3.4.2)
eNamespaceAttributeDriver1 #NamespaceAttribute ::= {
      ENCODE STRUCTURE {
          STRUCTURED WITH eNamespaceAttributePrepaddingAdder1 }
      WITH FastInfosetEncodingSet }

-- Кодирует единицу компонента attributes типа Element
--(см. C.3.6.1)
eAttributeDriver1 #Attribute ::= {
      ENCODE STRUCTURE {
          STRUCTURED WITH eAttributePrepaddingAdder1 }
      WITH FastInfosetEncodingSet }

-- Кодирует единицу 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 }
```

```
-- Кодирует единицу компонента element-name-surrogates и
-- attribute-name-surrogates типа Document (см. C.2.5.5)

eNameSurrogateDriver1 #NameSurrogate ::= {
      ENCODE STRUCTURE {
          STRUCTURED WITH eNameSurrogatePrepaddingAdder1 }
      WITH FastInfosetEncodingSet }

-- Кодирует компонент initial-vocabulary типа Document
--(см. C.2.5)

eInitialVocabularyPrepaddingAdder1 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eInitialVocabularyWithPrepadding1 }

-- Вставляет начальное дополнение перед кодированием единицы компонента
-- notations типа Document (см. C.2.6.1)

eNotationPrepaddingAdder1 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eNotationWithPrepadding1 }

-- Вставляет начальное дополнение перед кодированием единицы компонента
-- unparsed-entities типа Document (см. C.2.7.1)

eUnparsedEntityPrepaddingAdder1 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eUnparsedEntityWithPrepadding1 }

-- Вставляет начальное дополнение перед кодированием единицы компонента standalone
-- типа Document (см. C.2.9)

eStandalonePrepaddingAdder1 #BOOL ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eStandaloneWithPrepadding1 }

-- Вставляет начальное дополнение перед кодированием единицы компонента
-- children типа DocumentTypeDeclaration (см. C.9.6)

eDocTypeDeclChildPrepaddingAdder1 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eDocTypeDeclChildWithPrepadding1 }

-- Вставляет начальное дополнение перед кодированием единицы компонента
-- namespace-attributes типа Element (см. C.3.4.2)

eNamespaceAttributePrepaddingAdder1 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eNamespaceAttributeWithPrepadding1 }

-- Вставляет начальное дополнение перед кодированием единицы компонента
-- attributes типа Element (см. C.3.6.1)

eAttributePrepaddingAdder1 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eAttributeWithPrepadding1 }

-- Вставляет начальное дополнение перед кодированием единицы компонента
-- literal-qualified-name типа QualifiedNameOrIndex. Используется когда кодирование
-- начинается со второго бита октета (см. C.17.3)

eLiteralQualifiedNamePrepaddingAdder2 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eLiteralQualifiedNameWithPrepadding2 }

-- Вставляет начальное дополнение перед кодированием единицы компонента
-- literal-qualified-name типа QualifiedNameOrIndex. Используется когда кодирование
-- начинается с третьего бита октета (see C.18.3)

eLiteralQualifiedNamePrepaddingAdder3 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eLiteralQualifiedNameWithPrepadding3 }
```

```
-- Вставляет начальное дополнение перед кодированием единицы 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 }}

-- Вставляет начальное дополнение перед кодированием единицы 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 }

-- Вставляет начальное дополнение перед кодированием единицы компонентов
-- element-name-surrogates и attribute-name-surrogates типа Document
-- (см. C.2.5.5)

eNameSurrogatePrepaddingAdder1 #SEQUENCE ::= {
      REPLACE STRUCTURE WITH #PrecededByPrepadding
      ENCODED BY eNameSurrogateWithPrepadding1 }

-- Вставляет дискриминант перед кодированием единицы компонента
-- children типа Document (см. C.2.11.2 до C.2.11.5)

eDocumentChildDiscriminantAdder1or5 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
      ENCODED BY eDocumentChildWithDiscriminant1or5 }

-- Вставляет дискриминант перед кодированием единицы компонента
-- children типа Element (см. C.3.7.2 до C.3.7.6)

eElementChildDiscriminantAdder1or5 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByFiveAlternativeDiscriminant
      ENCODED BY eElementChildWithDiscriminant1or5 }

-- Вставляет дискриминант перед кодированием длины SEQUENCE OF,
-- идентифицирующий один из двух способов кодирования длины (см. C.21)

eNonEmptySequenceOfLengthDiscriminantAdder1 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByTwoAlternativeDiscriminant
      ENCODED BY eNonEmptySequenceOfLengthWithDiscriminant1 }

-- Вставляет дискриминант перед кодированием длины
-- NonEmptyOctetString, идентифицирующий один из трех способов
-- кодирования длины. Используется когда кодирование начинается со второго
-- бита октета (см. C.22)

eNonEmptyOctetStringLengthDiscriminantAdder2 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
      ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant2 }

-- Вставляет дискриминант перед кодированием длины
-- NonEmptyOctetString, идентифицирующий один из трех способов
-- кодирования длины. Используется когда кодирование начинается с пятого
-- бита октета (см. C.23)

eNonEmptyOctetStringLengthDiscriminantAdder5 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
      ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant5 }

-- Вставляет дискриминант перед кодированием длины
-- NonEmptyOctetString, идентифицирующий один из трех способов
-- кодирования длины. Используется когда кодирование начинается с седьмого
-- бита октета (см. C.24)

eNonEmptyOctetStringLengthDiscriminantAdder7 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
      ENCODED BY eNonEmptyOctetStringLengthWithDiscriminant7 }
```

```
-- Вставляет дискриминант перед кодированием положительного целого числа,
-- идентифицирующий один из трех способов кодирования такого числа.
-- Используется когда кодирование начинается со второго бита октета (см. C.25)
ePositiveIntegerDiscriminantAdder2 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByThreeAlternativeDiscriminant
      ENCODED BY ePositiveIntegerWithDiscriminant2 }

-- Вставляет дискриминант перед кодированием положительного целого числа,
-- идентифицирующий один из четырех способов кодирования такого числа.
-- Используется когда кодирование начинается с третьего бита октета (см. C.27)
ePositiveIntegerDiscriminantAdder3 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
      ENCODED BY ePositiveIntegerWithDiscriminant3 }

-- Вставляет дискриминант перед кодированием положительного целого числа,
-- идентифицирующий один из четырех способов кодирования такого числа.
-- Используется когда кодирование начинается с четвертого бита октета (см. C.28)
ePositiveIntegerDiscriminantAdder4 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
      ENCODED BY ePositiveIntegerWithDiscriminant4 }

-- Вставляет дискриминант перед кодированием неотрицательного целого числа,
-- идентифицирующий один из трех способов кодирования такого числа. (см. C.26)
eNonNegativeIntegerDiscriminantAdder2 #CHOICE ::= {
      REPLACE STRUCTURE WITH #PrecededByFourAlternativeDiscriminant
      ENCODED BY eNonNegativeIntegerWithDiscriminant2 }

-- Устанавливает начальное дополнение, добавленное перед компонентом
-- initial-vocabulary типа Document и кодирует этот компонент (см. C.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 }

-- Устанавливает начальное дополнение, добавленное перед каждой единицей
-- компонента notations типа Document и кодирует эту единицу (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 }

-- Устанавливает начальное дополнение, добавленное перед каждой единицей
-- компонента unparsed-entities типа Document и кодирует эту единицу
-- (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 }

-- Устанавливает начальное дополнение, добавленное перед компонентом
-- standalone типа Document и кодирует 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 }
```

```
-- Устанавливает начальное дополнение, добавленное перед каждой единицей
-- компонента namespace-attributes типа Element и кодирует эту единицу
-- (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 }

-- Устанавливает начальное дополнение, добавленное перед каждой единицей
-- компонента attributes типа Element и кодирует эту единицу
-- (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 }

-- Устанавливает начальное дополнение, добавленное перед каждой единицей
-- компонента children типа DocumentTypeDeclaration и кодирует эту единицу
-- (см. C.9.6)
eDocTypeDeclChildWithPrepadding1{<#C>} #PrecededByPrepadding{<#C>} ::= {
      ENCODE STRUCTURE {
          prepadding {
              ENCODING-SPACE SIZE 8
              PAD-PATTERN bits:'11100001'B },
          original eProcessingInstruction1 }
      WITH FastInfosetEncodingSet }
-- Устанавливает начальное дополнение, добавленное перед каждой единицей компонентов
-- restricted-alphabets, encoding-algorithms, prefixes, namespace-names,
-- local-names, other-ncnames и other-uris типа Document и кодирует эту
-- единицу (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 }
-- Устанавливает начальное дополнение, добавленное перед каждой единицей компонентов
-- attribute-values, content-character-chunks и other-strings
-- типа Document и кодирует эту единицу (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 }
```

```
-- Устанавливает начальное дополнение, добавленное перед компонентом
-- literal-qualified-name типа QualifiedNameOrIndex и кодирует этот
-- компонент. Используется 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 начальное дополнение, добавленное перед компонентом
-- literal-qualified-name типа QualifiedNameOrIndex и кодирует этот
-- компонент. Используется 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 }
-- Кодирует поле длины, добавленное перед SEQUENCE OF и кодирует
-- 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 }
-- Кодирует поле длины, добавленное перед SEQUENCE OF и кодирует
-- 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 }
```

```
-- Кодирует поле длины, добавленное перед SEQUENCE OF и кодирует
-- 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 }

-- Кодирует поле длины, добавленное перед SEQUENCE OF и кодирует
-- 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 }

-- Кодирует поле длины, добавленное перед NonEmptyOctetString и кодирует
-- NonEmptyOctetString. Используется, когда кодирование начинается со
-- второго бита октета (см. C.22)

eNonEmptyOctetStringWithLength2{<#C>}
#PrecededByNonEmptyOctetStringLength{<#C>} ::= {
      ENCODE STRUCTURE {
          length eNonEmptyOctetStringLength2,
          original eOctetStringOctets1{<length>} }
      WITH FastInfosetEncodingSet }

-- Кодирует поле длины, добавленное перед NonEmptyOctetString и кодирует
-- NonEmptyOctetString. Используется, когда кодирование начинается с
-- пятого бита октета (см. C.23)

eNonEmptyOctetStringWithLength5{<#C>}
      #PrecededByNonEmptyOctetStringLength{<#C>} ::= {
      ENCODE STRUCTURE {
          length eNonEmptyOctetStringLength5,
          original eOctetStringOctets1{<length>} }
      WITH FastInfosetEncodingSet }

-- Кодирует поле длины, добавленное перед NonEmptyOctetString и кодирует
-- NonEmptyOctetString. Используется, когда кодирование начинается с
-- седьмого бита октета (см. C.24)

eNonEmptyOctetStringWithLength7{<#C>}
#PrecededByNonEmptyOctetStringLength{<#C>} ::= {
      ENCODE STRUCTURE {
          length eNonEmptyOctetStringLength7,
          original eOctetStringOctets1{<length>} }
      WITH FastInfosetEncoingSet }

-- Кодирует дискриминант, добавленный перед единицей компонента children
-- типа Document и кодирует эту единицу (см. C.2.11.2 до C.2.11.5)

eDocumentChildWithDiscriminant1or5{<#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 }

-- Кодирует дискриминант, добавленный перед единицей компонента children
-- типа Element и кодирует эту единицу (см. C.3.7.2 до 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 }

-- Кодирует дискриминант, добавленный перед длиной SEQUENCE OF (идентифицирующий
-- один из двух способов кодирования длины), и кодирует длину (см. C.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 }
```

```
-- Кодирует дискриминант, добавленный перед длиной NonEmptyOctetString
-- (идентифицирующий один из трех способов кодирования длины), и кодирует длину.
-- Используется, когда кодирование начинается со второго бита октета (см. C.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 }

-- Кодирует дискриминант, добавленный перед длиной NonEmptyOctetString
-- (идентифицирующий один из трех способов кодирования длины), и кодирует длину.
-- Используется, когда кодирование начинается с пятого бита октета (см. C.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 }

-- Кодирует дискриминант, добавленный перед длиной NonEmptyOctetString
-- (идентифицирующий один из трех способов кодирования длины), и кодирует длину.
-- Используется, когда кодирование начинается с седьмого бита октета (см. C.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 }
```

```
-- Кодирует дискриминант, добавленный перед положительным целым числом
-- (идентифицирующий один из трех способов кодирования целого числа), и кодирует
-- это целое число. Используется, когда кодирование начинается со второго бита
-- октета (см. C.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}}}  -- Less than '1110'B
      WITH FastInfosetEncodingSet }

-- Кодирует дискриминант, добавленный перед положительным целым числом
-- (идентифицирующий один из четырех способов кодирования целого числа), и кодирует
-- это целое число. Используется, когда кодирование начинается с третьего бита
-- октета (см. C.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}}}  -- Less than '1111'B
      WITH FastInfosetEncodingSet }

-- Кодирует дискриминант, добавленный перед положительным целым числом
-- (идентифицирующий один из четырех способов кодирования целого числа), и кодирует
-- это целое число. Используется, когда кодирование начинается с четвертого бита
-- октета (см. C.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 }

-- Кодирует дискриминант, добавленный перед неотрицательным целым числом
-- (идентифицирующий один из трех способов кодирования этого целого числа), и
-- кодирует это целое число (см. C.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 }

-- Кодирует 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 }
```

```
-- Кодирует 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 }

-- Кодирует тип Attribute (см. C.4)
eAttribute2 #Attribute ::= {
      ENCODE STRUCTURE {
          qualified-name eQualifiedNameOrIndex2,
          normalized-value eNonIdentifyingStringOrIndex1 }
      WITH FastInfosetEncodingSet }

-- Кодирует тип ProcessingInstruction (см. C.5)
eProcessingInstruction1 #ProcessingInstruction ::= {
      ENCODE STRUCTURE {
          target eIdentifyingStringOrIndex1,
          content eNonIdentifyingStringOrIndex1 }
      WITH FastInfosetEncodingSet }

-- Кодирует тип UnexpandedEntityReference (см. C.6)
eUnexpandedEntityReference7 #UnexpandedEntityReference ::= {
      ENCODE STRUCTURE {
          name eIdentifyingStringOrIndex1,
          system-identifier eIdentifyingStringOrIndex1
                  OPTIONAL-ENCODING USE-SET,
          public-identifier eIdentifyingStringOrIndex1
                  OPTIONAL-ENCODING USE-SET }
      WITH FastInfosetEncodingSet }

-- Кодирует тип CharacterChunk (см. C.7)
eCharacterChunk3 #CharacterChunk ::= {
      ENCODE STRUCTURE {
          character-codes eNonIdentifyingStringOrIndex3 }
      WITH FastInfosetEncodingSet }

-- Кодирует тип Comment (см. C.8)
eComment1 #Comment ::= {
      ENCODE STRUCTURE {
          content eNonIdentifyingStringOrIndex1 }
      WITH FastInfosetEncodingSet }

-- Кодирует тип DocumentTypeDeclaration (см. C.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 }

-- Кодирует 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 }

-- Кодирует 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 }

-- Кодирует тип NamespaceAttribute (см. C.12)

eNamespaceAttribute7 #NamespaceAttribute ::= {
        ENCODE STRUCTURE {
            prefix eIdentifyingStringOrIndex1
                    OPTIONAL-ENCODING USE-SET,
            namespace-name eIdentifyingStringOrIndex1
                    OPTIONAL-ENCODING USE-SET }
        WITH FastInfosetEncodingSet }

-- Кодирует тип IdentifyingStringOrIndex (см. C.13)

eIdentifyingStringOrIndex1 #IdentifyingStringOrIndex ::= {
        ENCODE STRUCTURE {
            literal-character-string eNonEmptyOctetString2,
            string-index ePositiveInteger2 }
        WITH FastInfosetEncodingSet }

-- Кодирует тип NonIdentifyingStringOrIndex. Используется, 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 }

-- Кодирует тип NonIdentifyingStringOrIndex. Используется, когда кодирование
-- начинается с третьего бита октета (см. C.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 }

-- Кодирует тип NameSurrogate (см. C.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 }

-- Кодирует тип QualifiedNameOrIndex. Используется, когда кодирование
-- начинается со второго бита октета (см. C.17)

eQualifiedNameOrIndex2 #QualifiedNameOrIndex ::= {
      ENCODE STRUCTURE {
          literal-qualified-name {
              ENCODE STRUCTURE {
                  STRUCTURED WITH eLiteralQualifiedNamePrepaddingAdder2 }
              WITH FastInfosetEncodingSet },
          name-surrogate-index ePositiveInteger2
          STRUCTURED WITH eQualifiedNameAlternatives3 }
      WITH FastInfosetEncodingSet }

-- Кодирует тип QualifiedNameOrIndex. Используется, когда кодирование
-- начинается с третьего бита октета (см. C.18)

eQualifiedNameOrIndex3 #QualifiedNameOrIndex ::= {
      ENCODE STRUCTURE {
          literal-qualified-name {
              ENCODE STRUCTURE {
                  STRUCTURED WITH eLiteralQualifiedNamePrepaddingAdder3 }
              WITH FastInfosetEncodingSet },
          name-surrogate-index ePositiveInteger3
          STRUCTURED WITH eQualifiedNameAlternatives3 }
      WITH FastInfosetEncodingSet }

-- Кодирует тип EncodedCharacterString. Используется, когда кодирование
-- начинается с третьего бита октета (см. C.19)

eEncodedCharacterString3 #EncodedCharacterString ::= {
      ENCODE STRUCTURE {
          encoding-format USE-SET,
          octets eNonEmptyOctetString5 }
      WITH FastInfosetEncodingSet }

-- Кодирует тип EncodedCharacterString. Используется, когда кодирование
-- начинается с пятого бита октета (см. C.20)

eEncodedCharacterString5 #EncodedCharacterString ::= {
      ENCODE STRUCTURE {
          encoding-format USE-SET,
          octets eNonEmptyOctetString7 }
      WITH FastInfosetEncodingSet }

-- Кодирует повторение (SEQUENCE OF NonEmptyOctetString), вставляя перед ним
-- поле длины (см. C.2.5.3 до C.2.5.5)

eRepetitionWithLengthNonEmptyOctetString1 #REPETITION ::= {
      REPETITION-ENCODING {
          REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
          ENCODED BY eNonEmptySequenceOfWithLengthNonEmptyOctetString1 }}

-- Кодирует повторение (SEQUENCE OF EncodedCharacterString), вставляя перед ним
-- поле длины (см. C.2.5.3 до C.2.5.5)

eRepetitionWithLengthEncodedCharacterString1 #REPETITION ::= {
      REPETITION-ENCODING {
          REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
          ENCODED BY eNonEmptySequenceOfWithLengthEncodedCharacterString1 }}
```

```
-- Кодирует повторение (SEQUENCE OF NameSurrogate), вставляя перед ним
-- поле длины (см. C.2.5.3 до C.2.5.5)
eRepetitionWithLengthNameSurrogate1 #REPETITION ::= {
      REPETITION-ENCODING {
          REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
          ENCODED BY eNonEmptySequenceOfWithLengthNameSurrogate1 }}

-- Кодирует повторение (SEQUENCE OF additional-datum), вставляя перед ним
-- поле длины (см. C.2.5.3 до C.2.5.5)
eRepetitionWithLengthAdditionalDatum1 #REPETITION ::= {
      REPETITION-ENCODING {
          REPLACE STRUCTURE WITH #PrecededByNonEmptySequenceOfLength
          ENCODED BY eNonEmptySequenceOfWithLengthAdditionalDatum1 }}

-- Кодирует тип NonEmptyOctetString. Используется, когда кодирование начинается
-- со второго бита октета (см. C.22)
eNonEmptyOctetString2 #NonEmptyOctetString ::= {
      REPETITION-ENCODING {
          REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
          ENCODED BY eNonEmptyOctetStringWithLength2 }}

-- Кодирует тип NonEmptyOctetString. Используется, когда кодирование начинается
-- с пятого бита октета (см. C.23)
eNonEmptyOctetString5 #NonEmptyOctetString ::= {
      REPETITION-ENCODING {
          REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
          ENCODED BY eNonEmptyOctetStringWithLength5 }}

-- Кодирует тип NonEmptyOctetString. Используется, когда кодирование начинается
-- с седьмого бита октета (см. C.24)
eNonEmptyOctetString7 #NonEmptyOctetString ::= {
      REPETITION-ENCODING {
          REPLACE STRUCTURE WITH #PrecededByNonEmptyOctetStringLength
          ENCODED BY eNonEmptyOctetStringWithLength7 }}

-- Кодирует 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 }}

-- Кодирует поле длины, вставленное перед кодированием
-- NonEmptyOctetString. Используется, когда кодирование начинается
-- со второго бита октета (см. C.22)
eNonEmptyOctetStringLength2 #NonEmptyOctetStringLength ::= {
      USE #NonEmptyOctetStringLengthAlternatives2
      MAPPING ORDERED VALUES
      WITH {
          ENCODE STRUCTURE {
              STRUCTURED WITH eNonEmptyOctetStringLengthDiscriminantAdder2 }
          WITH FastInfosetEncodingSet }}

-- Кодирует поле длины, вставленное перед кодированием
-- NonEmptyOctetString. Используется, когда кодирование начинается
-- с пятого бита октета (см. C.23)
eNonEmptyOctetStringLength5 #NonEmptyOctetStringLength ::= {
      USE #NonEmptyOctetStringLengthAlternatives5
      MAPPING ORDERED VALUES
      WITH {
          ENCODE STRUCTURE {
              STRUCTURED WITH eNonEmptyOctetStringLengthDiscriminantAdder5 }
          WITH FastInfosetEncodingSet }}
```

```
-- Кодирует поле длины, вставленное перед кодированием
-- NonEmptyOctetString. Используется, когда кодирование начинается
-- с седьмого бита октета (см. C.24)

eNonEmptyOctetStringLength7 #NonEmptyOctetStringLength ::= {
      USE #NonEmptyOctetStringLengthAlternatives7
      MAPPING ORDERED VALUES
      WITH {
          ENCODE STRUCTURE {
              STRUCTURED WITH eNonEmptyOctetStringLengthDiscriminantAdder7 }
          WITH FastInfosetEncodingSet }}

-- Кодирует положительное целое число. Используется, когда кодирование начинается
-- со второго бита октета (см. C.25)

ePositiveInteger2 #PositiveOrNonNegativeInteger ::= {
      USE #PositiveIntegerAlternatives2
      MAPPING ORDERED VALUES
      WITH {
          ENCODE STRUCTURE {
              STRUCTURED WITH ePositiveIntegerDiscriminantAdder2 }
          WITH FastInfosetEncodingSet }}

-- Кодирует положительное целое число. Используется, когда кодирование начинается
-- с третьего бита октета (см. C.27)

ePositiveInteger3 #PositiveOrNonNegativeInteger ::= {
      USE #PositiveIntegerAlternatives3
      MAPPING ORDERED VALUES
      WITH {
          ENCODE STRUCTURE {
              STRUCTURED WITH ePositiveIntegerDiscriminantAdder3 }
          WITH FastInfosetEncodingSet }}

-- Кодирует положительное целое число. Используется, когда кодирование начинается
-- с четветрого бита октета (см. C.28)

ePositiveInteger4 #PositiveOrNonNegativeInteger ::= {
      USE #PositiveIntegerAlternatives4
      MAPPING ORDERED VALUES
      WITH {
          ENCODE STRUCTURE {
              STRUCTURED WITH ePositiveIntegerDiscriminantAdder4 }
          WITH FastInfosetEncodingSet }}

-- Кодирует неотрицательное целое число (см. C.26)

eNonNegativeInteger2 #PositiveOrNonNegativeInteger ::= {
      USE #NonNegativeIntegerAlternatives2
      MAPPING ORDERED VALUES
      WITH {
          ENCODE STRUCTURE {
              STRUCTURED WITH eNonNegativeIntegerDiscriminantAdder2 }
          WITH FastInfosetEncodingSet }}

-- Опеределяет, как устанавливать присутствие компонента namespace-attributes
-- типа Element (см. C.3.4.2)

eNamespaceAttributesOptionality3 #OPTIONAL ::= {
      PRESENCE DETERMINED BY handle
      HANDLE "nsa" }

-- Опеределяет, как устанавливать вариант типа QualifiedNameOrIndex
-- (см. C.17.3 и C.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}}   -- Less than '110011'B

-- Определяет, как установить окончание повторения, используя 4-битный
-- указатель конца '1111'(см. C.2.12, C.3.6.2, C.3.8, и C.9.7)
```

```
eRepetitionWithTerminator4bit1 #REPETITION ::= {
      REPETITION-ENCODING {
          REPETITION-SPACE SIZE variable-with-determinant
          DETERMINED BY pattern PATTERN bits:'1111'B }}

-- Определяет, как установить окончание повторения, используя 8-битный
-- указатель конца '11110000'(см. C.2.6.2 и C.2.7.2)

eRepetitionWithTerminator8bit1 #REPETITION ::= {
      REPETITION-ENCODING {
          REPETITION-SPACE SIZE variable-with-determinant
          DETERMINED BY pattern PATTERN bits:'11110000'B }}

-- Определяет, как установить окончание повторения, используя 10-битный
-- указатель конца '1111000000'(см. C.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 }}

-- Кодирует единицы SEQUENCE OF, следующие за добавленным полем длины
-- (см. C.2.5.3 до C.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 }}

-- Кодирует октеты NonEmptyOctetString, следующие за добавленным полем длины
-- (см. C.22, C.23 и C.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 B

Тип среды MIME media для документов Быстрого информационного набора

(Данное Приложение является неотъемлемой частью настоящей Рекомендации | Международного стандарта)

В данном Приложении определяется тип среды "application/fastinfoset", описывающий документы Быстрого информационного набора.

Этот тип MIME определяется ниже с использованием регистрационного шаблона MIME IETF и зарегистрирован в соответствии с процедурами IETF.

Имя типа среды MIME:
application

Имя подтипа MIME:
fastinfoset

Обязательные параметры:
Нет.

Необязательные параметры:
Нет.

Соображения, связанные с кодированием:
Информационные наборы XML, закодированные как документы Быстрого информационного набора, представляют собой двоичные данные. Для данного типа MIME может потребоваться последующее кодирование в случае использования транспортов, не поддерживающих двоичные данные.

Соображения безопасности:
Поскольку Информационные наборы XML, закодированные как документы Быстрого информационного набора, могут нести в себе данные, определенные приложением, семантика которых не зависит от семантики оболочки MIME (или контекста, в котором используется оболочка MIME), не следует полагать, что семантику документа Быстрого информационного набора можно будет понять на основе изучения одной только оболочки MIME. В связи с этим, во всех случаях использования типа среды "application/fastinfoset", настоятельно рекомендуется иметь полное понимание связанных с безопасностью последствий использования контекста, в котором применяется документ Быстрого информационного набора.

Соображения взаимодействия:
Известных проблем взаимодействия нет.

Опубликованная спецификация:
Рек. МСЭ-Т X.891 | ИСО/МЭК 24824-1

Приложения, использующие данный тип среды:
В настоящее время данный тип среды не используется никакими приложениями.

Дополнительная информация:

Магические числа:
Документ Быстрого информационного набора может начинаться с необязательной декларации XML, которая должна представлять собой одну 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 представляют собой шестнадцатеричную последовательность 3C 3F 78 6D 6C. Четыре октета, идентифицирующие документ Быстрого информационного набора, соответствующие подстроке "finf", закодированной в UTF-8, представляют собой шестнадцатеричную последовательность 66 69 6E 66.

Документ Быстрого информационного набора должен начинаться с шестнадцатеричной последовательности октетов E0 00 00 01, в случае, если необязательная декларация XML отсутствует.

Расширение(я) файлов:
*.finf

Контактное лицо и адрес электронной почты для получения дополнительной информации:
Докладчик по ASN.1 МСЭ-Т (адрес электронной почты: tsbmail@itu.int)
Докладчик по ASN.1 ИСО/МЭК JTC1/SC6 (адрес электронной почты: ittf@iso.org)

Возможное применение:
COMMON

Автор/Ответственный за изменения:
Совместные процедуры голосования МСЭ-Т | ИСО/МЭК в соответствии с Рек. МСЭ-Т A.23 *Сотрудничество с Международной организацией по стандартизации (ISO) и Международной электротехнической комиссией (IEC) по вопросам информационных технологий*, Приложение A и Директивы ИСО/МЭК JTC1, Приложение K.

Приложение C

Описание кодирования документа Быстрого информационного набора

(Данное Приложение не является неотъемлемой частью настоящей Рекомендации | Международного стандарта)

C.1 Документ Быстрого информационного набора

C.1.1 В данном Приложении неформально (но точно и полно) описываются кодирования, определенные в пункте 12 и Приложении A. Из соображений удобства все определения типов ASN.1, приведенные в основном тексте, скопированы в данное Приложение вместо использования ссылок на них.

C.1.2 Кодирования описаны в терминах действий, осуществляемых кодировщиком, в результате которых биты добавляются к потоку битов. Действия, которые должны выполняться декодером в данном Приложении явно не описываются, однако могут быть выведены из описанных в данном Приложении действий кодировщика.

C.1.3 Документ Быстрого информационного набора может начинаться либо с декларации XML (см. п. 12.3), за которой следуют:

a) шестнадцать битов '1110000000000000' (идентификация); за которыми следуют

b) шестнадцать битов '0000000000000001' (номер версии); за которыми следует

c) бит '0' (дополнение),

или с этих же тридцати трех битов, перед которыми нет декларации XML. Срезу же за тридцатью тремя битами следует кодирование абстрактного значения типа **Document**, как описано в C.2. Это кодирование заканчивается либо на восьмом, либо на четвертом бите октета, в зависимости от содержания документа Быстрого информационного набора. В последнем случае к потоку битов присоединяются четыре бита '0000' (дополнение).

C.2 Кодирование типа Document

C.2.1 Тип **Document** определен в п. 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
                  -- 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 }}
```

C.2.2 Значение типа **Document** кодируется путем выполнения приведенных ниже действий (в указанном порядке).

ПРИМЕЧАНИЕ. – Кодирование данного типа всегда начинается со второго бита октета и заканчивается либо на четвертом, либо на восьмом бите другого октета (являющемся последним битом указателя конца '1111', описанным в п. C.2.12).

C.2.3 Для каждого из семи необязательных компонентов **additional-data**, **initial-vocabulary**, **notations**, **unparsed-entities**, **character-encoding-scheme**, **standalone**, и **version** (в указанном порядке), если компонент присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.2.4 Если необязательный компонент **additional-data** присутствует, то количество компонентов **additional-datum** кодируется, как описано в п. C.21, и каждый из компонентов **additional-datum** кодируется, как описано в двух последующих подпунктах.

C.2.4.1 Бит '0' (дополнение) присоединяется к потоку битов и компонент **id** кодируется, как описано в п. C.22.

C.2.4.2 Бит '0' (дополнение) присоединяется к потоку битов и компонент **data** кодируется, как описано в п. C.22.

C.2.5 Если необязательный компонент **initial-vocabulary** присутствует, то три бита '000' (дополнение) присоединяется к потоку битов и этот компонент кодируется, как описано в пяти последующих подпунктах.

C.2.5.1 Для каждого из тринадцати необязательных компонентов **initial-vocabulary** (в порядку следования по тексту), если компонент присутствует, что к потоку битов присоединяется бит '1' (присутствие); в противном случае присоединяется бит '0' (отсутствие).

C.2.5.2 Если необязательны компонент **external-vocabulary** из **initial-vocabulary** присутствует, то к потоку битов добавляется бит '0' (дополнение), и компонент кодируется, как описано в п. C.22.

C.2.5.3 Для каждого из компонентов **restricted-alphabets**, **encoding-algorithms**, **prefixes**, **namespace-names**, **local-names**, **other-ncnames**, и **other-uris** (в указанном порядке), который присутствует, количество единиц **NonEmptyOctetString** в компоненте кодируется, как описано в п. C.21, и затем каждая единица (по порядку) кодируется следующим образом: К потоку битов присоединяется бит '0' (дополнение), и **NonEmptyOctetString** кодируется, как описано в п. C.22.

C.2.5.4 Для каждого из компонентов **attribute-values**, **content-character-chunks**, и **other-strings** (в указанном порядке), который присутствует, количество единиц **EncodedCharacterString** в компоненте кодируется, как описано в п. C.21, и затем каждая единица (по порядку) кодируется следующим образом: К потоку битов присоединяются два бита '00' (дополнение), и **EncodedCharacterString** кодируется, как описано в п. C.19.

C.2.5.5 Для каждого из компонентов **element-name-surrogates** и **attribute-name-surrogates** (в указанном порядке), который присутствует, количество единиц **NameSurrogate** в компоненте кодируется, как описано в п. C.21, и затем каждая единица (по порядку) кодируется следующим образом: К потоку битов присоединяются шесть битов '000000' (дополнение) **NameSurrogate** кодируется, как описано в п. C.16.

C.2.6 Если необязательный компонент **notations** присутствует, то он кодируется, как описано в двух последующих подпунктах.

C.2.6.1 Каждая единица **notations** (по порядку) кодируется следующим образом: К потоку битов присоединяются шесть битов '110000' (идентификация) и **Notation** кодируется, как описано в п. C.11.

C.2.6.2 К потоку битов присоединяются четыре бита '1111' (указатель конца) и четыре бита '0000' (дополнение).

ПРИМЕЧАНИЕ. – Эти биты не присоединяются, если компонент **notations** отсутствует.

C.2.7 Если необязательный компонент **unparsed-entities** присутствует, то он кодируется, как описано в двух последующих подпунктах.

C.2.7.1 Каждая единица **unparsed-entities** (по порядку) кодируется следующим образом: К потоку битов присоединяются семь битов '1101000' (идентификация) и **UnparsedEntity** кодируется, как описано в п. C.10.

C.2.7.2 К потоку битов присоединяются четыре бита '1111' (указатель конца) и четыре бита '0000' (дополнение).

ПРИМЕЧАНИЕ. – Эти биты не присоединяются, если компонент **unparsed-entities** отсутствует.

C.2.8 Если необязательный компонент **character-encoding-scheme** присутствует, то к потоку битов присоединяется бит '0' (дополнение), и **NonEmptyOctetString** кодируется, как описано в п. C.22.

C.2.9 Если необязательный компонент **standalone** присутствует, то он кодируется следующим образом: К потоку битов присоединяются семь битов '0000000' (дополнение). Если **standalone** имеет значение **TRUE**, то к потоку битов присоединяется бит '1', иначе присоединяется бит '0'.

C.2.10 Если необязательный компонент **version** присутствует, то его значение кодируется, как описано в п. C.14.

C.2.11 Если компонент **children** имеет одну или более единиц, то каждая из этих единиц кодируется (по порядку), как описано в последующих пяти подпунктах.

C.2.11.1 Требуется, чтобы кодирование каждой единицы начиналось с первого бита октета. Однако последний присоединенный бит мог быть либо восьмым, либо четвертым битом октета. Если он был четвертым битом октета, к потоку битов присоединяются биты '0000' (дополнение) с тем, чтобы кодирование единицы начиналось с первого бита следующего октета.

C.2.11.2 Если присутствует вариант **element**, то к потоку битов присоединяется бит '0' (идентификация) и **element**, затем, кодируется, как описано в п. C.3.

C.2.11.3 Если присутствует вариант **processing-instruction**, то к потоку битов присоединяются восемь бит '11100001' (идентификация) и **processing-instruction** кодируется, как описано в п. C.5.

C.2.11.4 Если присутствует вариант **comment**, то к потоку битов присоединяются восемь битов '11100010' (идентификация) и **comment** кодируется, как описано в п. C.8.

C.2.11.5 Если присутствует вариант **document-type-declaration** , то к потоку битов присоединяются шесть бит '110001' (идентификация) и **document-type-declaration** кодируется, как описано в п. C.9.

C.2.12 Присоединяются четыре бита '1111' (указатель конца).

ПРИМЕЧАНИЕ. – Эти биты не присоединяются, если компонент **children** не имеет единиц.

C.3 Кодирование типа Element

C.3.1 Тип **Element** определен в п. 7 .3следующим 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 }}
```

C.3.2 Значение типа **Element** кодируется путем выполнения следующих действий (по порядку).

ПРИМЕЧАНИЕ. – Кодирование этого типа всегда начинается со второго бита октета и заканчивается либо на четвертом, либо на восьмом бите другого октета (являющемся последним битом указателя конца '1111', описанного в п. C.3.8).

C.3.3 Если необязательный компонент **attributes** присутствует, то бит '1' (присутствие) присоединяется к потоку битов, в противном случае присоединяется бит '0' (отсутствие).

C.3.4 Если необязательный компонент **namespace-attributes** присутствует, то он кодируется, как описано в трех последующих подпунктах.

C.3.4.1 К потоку битов присоединяются четыре бита '1110' (присутствие) и два бита '00' (дополнение).

C.3.4.2 Каждая единица **namespace-attributes** (по порядку) кодируется следующим образом: К потоку битов присоединяются шесть битов '110011' (идентификация), и **NamespaceAttribute** кодируется, как описано в п. C.12.

C.3.4.3 Присоединяются четыре бита '1111' (указатель конца) и шесть битов '000000' (дополнение).

ПРИМЕЧАНИЕ. – Эти биты не присоединяются, если компонент **namespace-attributes** отсутствует.

C.3.5 Значение компонента **qualified-name** кодируется, как описано в п. C.18.

C.3.6 Если необязательный компонент **attributes** присутствует, то он кодируется, как описано в двух последующих подпунктах.

C.3.6.1 Каждая единица **attributes** (по порядку) кодируется следующим образом: К потоку битов присоединяется бит '0' (идентификация), и **Attribute** кодируется, как описано в п.C.4.

C.3.6.2 Присоединяются четыре бита '1111' (указатель конца).

ПРИМЕЧАНИЕ. – Эти биты не присоединяются, если компонент **attributes** отсутствует.

C.3.7 Если компонент **children** имеет одну или более единиц, то каждая единица кодируется (по порядку), как описано в шести последующих подпунктах.

C.3.7.1 Требуется, чтобы кодирование каждой единицы начиналось с первого бита октета. Однако последний присоединенный бит мог быть либо восьмым, либо четвертым битом октета. Если он был четвертым битом октета, к потоку битов присоединяются биты '0000' (дополнение) с тем, чтобы кодирование единицы начиналось с первого бита следующего октета.

C.3.7.2 Если присутствует вариант **element** то к потоку битов присоединяется бит '0' (идентификация) и **element**, затем, кодируется, как описано в п. C.3.

C.3.7.3 Если присутствует вариант **processing-instruction**, то к потоку битов присоединяются восемь бит '11100001' (идентификация) и **processing-instruction** кодируется, как описано в п. C.5.

C.3.7.4 Если присутствует вариант **unexpanded-entity-reference**, то к потоку битов присоединяются шесть бит '110010' (идентификация), и **unexpanded-entity-reference** кодируется, как описано в п. C.6.

C.3.7.5 Если присутствует вариант **character-chunk** то к потоку битов присоединяются два бита '10' (идентификация), и **character-chunk** кодируется, как описано в п C.7.

C.3.7.6 Если присутствует вариант **comment**, то к потоку битов присоединяются восемь битов '11100010' (идентификация) и **comment** кодируется, как описано в п. C.8.

C.3.8 Присоединяются четыре бита '1111' (указатель конца).

ПРИМЕЧАНИЕ. – Эти биты не присоединяются, если компонент **children** не имеет единиц.

C.4 Кодирование типа Attribute

C.4.1 Тип **Attribute** определен в п. 7.4 следующим образом:

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

C.4.2 Значение типа **Attribute** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается со второго бита октета и заканчивается на восьмом бите другого октета.

C.4.3 Значение **qualified-name** кодируется, как описано в п. C.17.

C.4.4 Значение **normalized-value** кодируется, как описано в п. C.14.

C.5 Кодирование типа ProcessingInstruction

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

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

C.5.2 Значение типа **ProcessingInstruction** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с первого бита октета и заканчивается на восьмом бите другого октета.

C.5.3 Значение **target** кодируется, как описано в п.C.13.

C.5.4 Значение **content** кодируется, как описано в п. C.14.

C.6 Кодирование типа UnexpandedEntityReference

C.6.1 Тип **UnexpandedEntityReference** определен в п. 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** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с седьмого бита октета и заканчивается на восьмом бите другого октета.

C.6.3 Для каждого из необязательных компонентов **system-identifier** и **public-identifier** (в указанном порядке), если компонент присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.6.4 Значение **name** кодируется, как описано в п. C.13.

C.6.5 Если необязательный компонент **system-identifier** присутствует, то он кодируется, как описано в п. C.13.

C.6.6 Если необязательный компонент **public-identifier** присутствует, то он кодируется, как описано в п. C.13.

C.7 Кодирование типа CharacterChunk

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

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

C.7.2 Значение типа **CharacterChunk** кодируется, путем выполнения следующего действия.

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с третьего бита октета и заканчивается на восьмом бите другого или того же октета.

C.7.3 Значение **character-codes** кодируется, как описано в п. C.15.

C.8 Кодирование типа Comment

C.8.1 Тип **Comment** определен в п. 7.8 следующим образом:

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

C.8.2 Значение типа **Comment** кодируется, путем выполнения следующего действия.

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с первого бита октета и заканчивается на восьмом бите другого или того же октета.

C.8.3 Значение **content** кодируется, как описано в п. C.14.

C.9 Кодирование типа DocumentTypeDeclaration

C.9.1 Тип **DocumentTypeDeclaration** определен в п. 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 }
```

C.9.2 Значение типа **DocumentTypeDeclaration** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с седьмого бита октета и заканчивается на четвертом бите другого октета (являющемся последним битом указателя конца '1111', описанного в п. 9.7).

C.9.3 Для каждого из необязательных компонентов **system-identifier** и **public-identifier** (в указанном порядке), если компонент присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.9.4 Если необязательный компонент **system-identifier** присутствует, то он кодируется, как описано в п. C.13.

C.9.5 Если необязательный компонент **public-identifier** присутствует, то он кодируется, как описано в п. C.13.

C.9.6 Если компонент **children** имеет одну или более единиц, то каждая из этих единиц кодируется следующим образом: К потоку битов присоединяются восемь битов '11100001' (идентификация), и **ProcessingInstruction** кодируется, как описано в п. C.5.

C.9.7 Присоединяются четыре бита '1111' (указатель конца).

ПРИМЕЧАНИЕ. – Эти биты не присоединяются, если компонент **children** не имеет единиц.

C.10 Кодирование типа UnparsedEntity

C.10.1 Тип **UnparsedEntity** определен в п. 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 -- }
```

C.10.2 Значение типа **UnparsedEntity** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с восьмого бита октета и заканчивается на восьмом бите другого октета.

C.10.3 Если необязательный компонент **public-identifier** присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.10.4 Значение **name** кодируется, как описано в п. C.13.

C.10.5 Значение **system-identifier** кодируется, как описано в п. C.13.

C.10.6 Если необязательный компонент **public-identifier** присутствует, то он кодируется, как описано в п. C.13.

C.10.7 Значение **name** кодируется, как описано в п. C.13.

C.11 Кодирование типа Notation

C.11.1 Тип **Notation** определен в п. 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 -- }
```

C.11.2 Значение типа **Notation** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с седьмого бита октета и заканчивается на восьмом бите другого октета.

C.11.3 Для каждого из необязательных компонентов **system-identifier** и **public-identifier** (в указанном порядке), если компонент присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.11.4 Значение **name** кодируется, как описано в п. C.13.

C.11.5 Если необязательный компонент **system-identifier** присутствует, то он кодируется, как описано в п. C.13.

C.11.6 Если необязательный компонент **public-identifier** присутствует, то он кодируется, как описано в п. C.13.

C.12 Кодирование типа NamespaceAttribute

C.12.1 Тип **NamespaceAttribute** определен в п. 7.12 следующим образом:

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

C.12.2 Значение типа **NamespaceAttribute** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с восьмого бита октета и заканчивается на восьмом бите другого октета.

C.12.3 Если необязательный компонент **prefix** присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.12.4 Если необязательный компонент **namespace-name** присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.12.5 Если необязательный компонент **prefix** присутствует, то он кодируется, как описано в п. C.13.

C.12.6 Если необязательный компонент **namespace-name** присутствует, то он кодируется, как описано в п. C.13.

C.13 Кодирование типа IdentifyingStringOrIndex

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

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

C.13.2 Значение типа **IdentifyingStringOrIndex** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с первого бита октета и заканчивается на восьмом бите другого или того же октета.

C.13.3 Если присутствует вариант **literal-character-string**, то к потоку битов присоединяется бит '0' (дискриминант), и **literal-character-string** кодируется, как описано в п. C.22.

C.13.4 Если присутствует вариант **string-index**, то к потоку битов присоединяется бит '1' (дискриминант) , и **string-index** кодируется, как описано в п. C.25

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

C.14.1 Тип **NonIdentifyingStringOrIndex** определен в п.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 подпункте C.14 описывается кодирование значения типа **NonIdentifyingStringOrIndex**, когда кодирование начинается с первого бита октета (см. также п. C.15). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого или того же октета.

C.14.3 Если присутствует вариант **literal-character-string**, то к потоку битов присоединяется бит '0' (дискриминант), и **literal-character-string** кодируется, как описано в двух последующих подпунктах.

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

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

C.14.4 Если присутствует вариант **string-index**, то к потоку битов присоединяется бит '1' (дискриминант) , и **string-index** кодируется, как описано в п. C.26.

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

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

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

C.15.2 В данном подпункте C.15 описывается кодирование значения типа **NonIdentifyingStringOrIndex**, когда кодирование начинается с третьего бита октета (см. также п. C.14). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого или того же октета.

C.15.3 Если присутствует вариант **literal-character-string**, то к потоку битов присоединяется бит '0' (дискриминант), и **literal-character-string** кодируется, как описано в двух последующих подпунктах.

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

C.15.3.2 Значение компонента **character-string** кодируется, как описано в п. C.20.

C.15.4 Если присутствует вариант **string-index**, то к потоку битов присоединяется бит '1' (дискриминант) , и **string-index** кодируется, как описано в п. C.28.

C.16 Кодирование типа NameSurrogate

C.16.1 Тип **NameSurrogate** определен в п. 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 --})
```

C.16.2 Значение типа **NameSurrogate** кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается с седьмого бита октета и заканчивается на восьмом бите другого октета.

C.16.3 Если необязательный компонент **prefix-string-index** присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.16.4 Если необязательный компонент **namespace-name-string-index** присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.16.5 Если необязательный компонент **prefix-string-index** присутствует, то к потоку битов присоединяется бит '0' (дополнение), и компонент кодируется, как описано в п. C.25.

C.16.6 Если необязательный компонент **namespace-name-string-index** присутствует, то к потоку битов присоединяется бит '0' (дополнение), и компонент кодируется, как описано в п. C.25.

C.16.7 К потоку битов присоединяется бит '0' (дополнение), и компонент **local-name-string-index** кодируется, как описано в п. C.25.

C.17 Кодирование типа QualifiedNameOrIndex, начинающееся со второго бита октета

C.17.1 Тип **QualifiedNameOrIndex** определен в п. 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) }
```

C.17.2 В данном подпункте C.17 описывается кодирование значения типа **QualifiedNameOrIndex**, когда кодирование начинается со второго бита октета (см. также п. C.18). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого или того же октета.

C.17.3 Если присутствует вариант **literal-qualified-name**, то к потоку битов присоединяются четыре бита '1111' (идентификация) и бит '0' (дополнение), и **literal-qualified-name** кодируется, как описано в четырех последующих подпунктах.

C.17.3.1 Для каждого из необязательных компонентов **prefix** и **namespace-name** (в указанном порядке), если компонент присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.17.3.2 Если присутствует необязательный компонент **prefix**, то он кодируется, как описано в п. C.13.

C.17.3.3 Если присутствует необязательный компонент **namespace-name**, то он кодируется, как описано в п. C.13.

C.17.3.4 Компонент **local-name** кодируется, как описано в п. C.13.

C.17.4 Если присутствует вариант **name-surrogate-index** , то он кодируется, как описано в п. C.25.

C.18 Кодирование типа QualifiedNameOrIndex, начинающееся с третьего бита октета

C.18.1 Тип **QualifiedNameOrIndex** определен в п. 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) }
```

C.18.2 В данном подпункте C.18 описывается кодирование значения типа **QualifiedNameOrIndex**, когда кодирование начинается с третьего бита октета (см. также п. C.17). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого или того же октета.

C.18.3 Если присутствует вариант **literal-qualified-name**, то к потоку битов присоединяются четыре бита '1111' (идентификация) и **literal-qualified-name** кодируется, как описано в четырех последующих подпунктах.

C.18.3.1 Для каждого из необязательных компонентов **prefix** и **namespace-name** (в указанном порядке), если компонент присутствует, то к потоку битов присоединяется бит '1' (присутствие), в противном случае присоединяется бит '0' (отсутствие).

C.18.3.2 Если присутствует необязательный компонент **prefix**, то он кодируется, как описано в п. C.13.

C.18.3.3 Если присутствует необязательный компонент **namespace-name**, то он кодируется, как описано в п. C.13.

C.18.3.4 Компонент **local-name** кодируется, как описано в п. C.13.

C.18.4 Если присутствует вариант **name-surrogate-index** , то он кодируется, как описано в п. C.27.

C.19 Кодирование типа EncodedCharacterString, начинающееся с третьего бита октета

C.19.1 Тип **EncodedCharacterString** определен в п. 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 }
```

C.19.2 В данном подпункте C.19 описывается кодирование значения типа **EncodedCharacterString**, когда кодирование начинается с третьего бита октета (см. также п. C.20). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого октета.

C.19.3 Значение компонента **encoding-format** кодируется, как описано в четырех последующих подпунктах.

C.19.3.1 Если присутствует вариант **utf-8**, то к потоку битов присоединяются два бита '00' (дискриминант).

C.19.3.2 Если присутствует вариант **utf-16**, то к потоку битов присоединяются два бита '01' (дискриминант).

C.19.3.3 Если присутствует вариант **restricted-alphabet** , то к потоку битов присоединяются два бита '10' (дискриминант), и **restricted-alphabet** кодируется, как описано в п. C.29.

C.19.3.4 Если присутствует вариант **encoding-algorithm** , то к потоку битов присоединяются два бита '11' (дискриминант), и **encoding-algorithm** кодируется, как описано в п. C.29.

C.19.4 Компонент **octets** кодируется, как описано в п. C.23.

C.20 Кодирование типа EncodedCharacterString, начинающееся с пятого бита октета

C.20.1 Тип **EncodedCharacterString** определен в п. 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 }
```

C.20.2 В данном подпункте C.20 описывается кодирование значения типа **EncodedCharacterString**, когда кодирование начинается с пятого бита октета (см. также п. C.19). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого октета.

C.20.3 Значение компонента **encoding-format** кодируется, как описано в четырех последующих подпунктах.

C.20.3.1 Если присутствует вариант **utf-8**, то к потоку битов присоединяются два бита '00' (дискриминант).

C.20.3.2 Если присутствует вариант **utf-16**, то к потоку битов присоединяются два бита '01' (дискриминант).

C.20.3.3 Если присутствует вариант **restricted-alphabet** , то к потоку битов присоединяются два бита '10' (дискриминант), и **restricted-alphabet** кодируется, как описано в п. C.29.

C.20.3.4 Если присутствует вариант **encoding-algorithm** , то к потоку битов присоединяются два бита '11' (дискриминант), и **encoding-algorithm** кодируется, как описано в п. C.29.

C.20.4 Компонент **octets** кодируется, как описано в п. C.24.

C.21 Кодирование длины типа sequence-of

C.21.1 В данном подпункте описывается кодирование длины типа sequence-of, закодированного с полем длины, предшествующим единицам типа sequence-of.

ПРИМЕЧАНИЕ. – Данное кодирование всегда начинается с первого бита октета и заканчивается восьмым битого другого или того же октета.

C.21.2 Если значение находится в диапазоне от 1 до 128, то к потоку битов присоединяется бит '0' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из семи битов и присоединяется к потоку битов.

C.21.3 Если значение находится в диапазоне от 129 до 2^{20}, то к потоку битов присоединяются бит '1' и три бита '000' (дополнение), и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из двадцати битов и присоединяется к потоку битов.

C.22 Кодирование типа NonEmptyOctetString, начинающееся со второго бита октета

C.22.1 Тип **NonEmptyOctetString** определен в п. 7.2 следующим образом:

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

C.22.2 В данном подпункте C.22 описывается кодирование значения типа **NonEmptyOctetString**, когда кодирование начинается со второго бита октета (см. также п. C.23 и п. C.24). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого октета.

C.22.3 Длина строки октетов кодируется, как описано в трех последующих подпунктах.

C.22.3.1 Если длина находится в диапазоне от 1 до 64, то к потоку битов присоединяется бит '0' и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из шести битов и присоединяется к потоку битов.

C.22.3.2 Если длина находится в диапазоне от 65 до 320, то к потоку битов присоединяются два бита '10' и пять битов '00000' (дополнение) и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из восьми битов и присоединяется к потоку битов.

C.22.3.3 Если длина находится в диапазоне от 321 до 2^{32}, то к потоку битов присоединяются два бита '11' и пять битов '00000' (дополнение) и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из тридцати двух битов и присоединяется к потоку битов.

C.22.4 Биты, формирующие октеты строки октетов, присоединяются к потоку битов (по порядку).

C.23 Кодирование типа `NonEmptyOctetString`, начинающееся с пятого бита октета

C.23.1 Тип **`NonEmptyOctetString`** определен в п. 7.2 следующим образом:

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

C.23.2 В данном подпункте C.23 описывается кодирование значения типа **`NonEmptyOctetString`**, когда кодирование начинается с пятого бита октета (см. также п. C.22 и п. C.24). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого октета.

C.23.3 Длина строки октетов кодируется, как описано в трех последующих подпунктах.

C.23.3.1 Если длина находится в диапазоне от 1 до 8, то к потоку битов присоединяется бит '0' и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из трех битов и присоединяется к потоку битов.

C.23.3.2 Если длина находится в диапазоне от 9 до 265, то к потоку битов присоединяются два бита '10' и два бита '00' (дополнение) и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из восьми битов и присоединяется к потоку битов.

C.23.3.3 Если длина находится в диапазоне от 265 до 2^{32}, то к потоку битов присоединяются два бита '11' и два бита '00' (дополнение) и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из тридцати двух битов и присоединяется к потоку битов.

C.23.4 Биты, формирующие октеты строки октетов, присоединяются к потоку битов (по порядку).

C.24 Кодирование типа `NonEmptyOctetString`, начинающееся с седьмого бита октета

C.24.1 Тип **`NonEmptyOctetString`** определен в п. 7.2 следующим образом:

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

C.24.2 В данном подпункте C.24 описывается кодирование значения типа **`NonEmptyOctetString`**, когда кодирование начинается с седьмого бита октета (см. также п. C.22 и п. C.23). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого октета.

C.24.3 Длина строки октетов кодируется, как описано в трех последующих подпунктах.

C.24.3.1 Если длина находится в диапазоне от 1 до 2, то к потоку битов присоединяется бит '0' и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из одного бита и присоединяется к потоку битов.

C.24.3.2 Если длина находится в диапазоне от 3 до 258, то к потоку битов присоединяются два бита '10' и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из восьми битов и присоединяется к потоку битов.

C.24.3.3 Если длина находится в диапазоне от 258 до 2^{32}, то к потоку битов присоединяются два бита '11' и длина, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из тридцати двух битов и присоединяется к потоку битов.

C.24.4 Биты, формирующие октеты строки октетов, присоединяются к потоку битов (по порядку).

C.25 Кодирование целых чисел из диапазона от 1 до 2^{20}, начинающееся со второго бита октета

C.25.1 В данном подпункте C.25 описывается кодирование целочисленных значений из диапазона 1 до 2^{20}, когда кодирование начинается со второго бита октета (см. также п. 26, п. C.27 и п. C.28). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого октета.

C.25.2 Если значение находится в диапазоне от 1 до 64, то к потоку битов присоединяется бит '0' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из шести битов и присоединяется к потоку битов.

C.25.3 Если значение находится в диапазоне от 65 до 8256, то к потоку битов присоединяются два бита '10' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из тринадцати битов и присоединяется к потоку битов.

C.25.4 Если значение находится в диапазоне от 8257 до 2^{20}, то к потоку битов присоединяются два бита '10' и бит '0' (дополнение) и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из двадцати битов и присоединяется к потоку битов.

C.26 Кодирование целых чисел в диапазоне от 0 до 2^{20}, начинающееся со второго бита октета

C.26.1 В данном подпункте C.26 описывается кодирование целочисленных значений из диапазона 0 до 2^{20}, когда кодирование начинается со второго бита октета (см. также п. 25, п. C.27 и п. C.28). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого или этого же октета.

C.26.2 Если значение равно нулю, то к потоку битов присоединяются семь битов '1111111'. В противном случае значение кодируется, как описано в п. C.25.

C.27 Кодирование целых чисел в диапазоне от 1 до 2^{20}, начинающееся с третьего бита октета

C.27.1 В данном подпункте C.27 описывается кодирование целочисленных значений из диапазона 1 до 2^{20}, когда кодирование начинается с третьего бита октета (см. также п. 25, п. C.26 и п. C.28). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого октета.

C.27.2 Если значение находится в диапазоне от 1 до 32, то к потоку битов присоединяется бит '0' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из пяти битов и присоединяется к потоку битов.

C.27.3 Если значение находится в диапазоне от 33 до 2080, то к потоку битов присоединяются три бита '100' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из одиннадцати битов и присоединяется к потоку битов.

C.27.4 Если значение находится в диапазоне от 2081 до 526368, то к потоку битов присоединяются три бита '101' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из девятнадцати битов и присоединяется к потоку битов.

C.27.5 Если значение находится в диапазоне от 526369 до 2^{20}, то к потоку битов присоединяются три бита '110' и семь битов '0000000' (дополнение) и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из двадцати битов и присоединяется к потоку битов.

C.28 Кодирование целых чисел в диапазоне от 1 до 2^{20}, начинающееся с четвертого бита октета

C.28.1 В данном подпункте C.28 описывается кодирование целочисленных значений из диапазона 1 до 2^{20}, когда кодирование начинается с четвертого бита октета (см. также п. 25, п. C.26 и п. C.27). Данное значение кодируется, путем выполнения следующих действий (в указанном порядке).

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда заканчивается на восьмом бите другого или того же октета.

C.28.2 Если значение находится в диапазоне от 1 до 16, то к потоку битов присоединяется бит '0' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из четырех битов и присоединяется к потоку битов.

C.28.3 Если значение находится в диапазоне от 17 до 1040, то к потоку битов присоединяются три бита '100' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из десяти битов и присоединяется к потоку битов.

C.28.4 Если значение находится в диапазоне от 1041 до 263184, то к потоку битов присоединяются три бита '101' и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из восемнадцати битов и присоединяется к потоку битов.

C.28.5 Если значение находится в диапазоне от 263185 до 2^{20}, то к потоку битов присоединяются три бита '100' и шесть битов '000000' (дополнение), и значение, за вычетом нижней границы диапазона, кодируется как целое число без знака в поле из двадцати битов и присоединяется к потоку битов.

C.29 Кодирование целых чисел в диапазоне от 1 до 256

C.29.1 В данном подпункте C.28 описывается кодирование целочисленных значений из диапазона 1 до 256.

ПРИМЕЧАНИЕ. –Кодирование данного типа всегда начинается либо с пятого, либо с седьмого бита октета и заканчивается либо на четвертом, либо на шестом бите (соответственно) следующего октета.

C.29.2 Значение, за вычетом нижней границы интервала, кодируется как целое число без знака в поле из восьми битов и присоединяется к потоку битов.

Приложение D

Примеры кодирования Информационных наборов XML как документов Быстрого информационного набора

(Данное Приложение не является неотъемлемой частью настоящей Рекомендации | Международного стандарта)

D.1 Представление примеров

D.1.1 В данном приложении используются следующие соглашения об обозначениях для чисел:

a) для цифр числа, представленного в десятичной системе счисления (десятичное число), используется **`полужирное начертание гарнитурой Courier`**, за которым следует нижний индекс "10" (например, $\mathbf{11}_{10}$);

b) для цифр числа представленного в шестнадцатеричной системе счисления, (шестнадцатеричное число) используется **`полужиное начертание гарнитурой Courier`**, за которым следует нижний индекс "10" (например, $\mathbf{0b1f}_{16}$);

c) если система счисления числа указана явным образом, нижний индекс опускается.

D.1.2 В данном Приложении представлено два примера возможных кодирований заказа, оформленного с помощью Универсального языка бизнеса (UBL) [1], в документ Быстрого информационного набора. UBL разработан для предоставления универсально понятного и общепризнанного коммерческого синтаксиса для обязательных юридических документов.

D.1.3 Информационный набор XML для примера UBL представлен в п. D.3.

D.1.4 Исходный словарь первого документа Быстрого информационного набора ссылается на внешний словарь. В подпункте D.4 описывается содержание внешнего словаря, октеты документа Быстрого информационного набора и приводятся объяснения некоторых последовательностей октетов.

D.1.5 Второй Быстрый информационный набор не имеет исходного словаря. В подпункте D.5 описываются октеты документа Быстрого информационного набора и приводятся объяснения некоторых последовательностей октетов.

ПРИМЕЧАНИЕ. – Окончательный словарь для этого документа Быстрого информационного набора аналогичен окончательному словарю для документа Быстрого информационного набора, описанного в п. D.4.

D.1.6 Октеты п. D.4 и D.5 представлены двумя столбцами в ряде таблиц. В первом столбце перечислены в шестнадцатеричном виде начальные позиции 32-х последовательно идущих октетов документа Быстрого информационного набора, а во втором столбце перечислены эти октеты в шестнадцатеричной нотации. Подчеркиванием выделены шестнадцатеричные символы, содержащие биты, соответствующие идентификации и указателю конца единиц информации.

D.1.7 Объяснения некоторых последовательностей октетов документов Быстрого информационного набора (в п. D.4 и D.5) представлены в следующих столбцах:

a) В столбце 1 представлена, в шестнадцатеричном виде, позиция октета(октетов), перечисленных в столбце 2.

b) В столбце 2 представлен октет(октеты) документа Быстрого информационного набора, связанные с соответствующей единице информации и свойствами этой единицы. Октет представлен в двоичной системе счисления, за двоичным представлением в скобках следует шестнадцатеричное представление того же октета (например, **`11110000 (f0)`**).

c) В столбце 3 приводится детальное описание октета, приведенного в столбце 2, и даются ссылки на подпункты Приложения C, к которым следует обращаться за дальнейшими разъяснениями.

d) В столбце 4 содержится часть Информационного набора XML или часть документа XML 1.0 (если он применим), соответствующая октету(октетам), приведенным в столбце 2.

D.1.8 В этих примерах все блоки единиц информации **character**, содержащие менее 6 символов, добавляются в таблицу CONTENT CHARACTER CHUNK, и значения свойства **[normalized value]** всех единиц информации **attribute**, содержащие менее 6 символов, добавляются в таблицу ATTRIBUTE VALUE.

D.1.9 Размеры документов XML 1.0 и документов Быстрого информационного набора, а также размеры этих документов, сжатых методом GZIP, приведены в п. D.2.

D.2 Размер документов-примеров (включая сжатие на основе избыточности)

D.2.1 В Таблице D.1 приведены размеры всех документов. В столбце 1 приведены документы UBL, в столбце 2 приведены размеры документов, а в столбце 3 приведены размеры документов, сжатых методом GZIP [2] (с установками по умолчанию).

ПРИМЕЧАНИЕ 1. – Документ Заказ UBL XML 1.0 не содержит пробельных символов (см. п. D.3.1.2).

ПРИМЕЧАНИЕ 2. – В каждом из документов все символы закодированы с использованием кодирования символов UTF-8.

ПРИМЕЧАНИЕ 3. – Для документов Быстрого информационного набора не сериализовано декларации XML (см. п. 12.3).

Таблица D.1 – Исходные размеры документов и размеры документов после сжатия методом GZIP

Документ UBL	Размер	Размер после сжатия GZIP
Документа XML 1.0	3311	893
Документ Быстрого информационного набора с внешним словарем	684	546
Документ Быстрого информационного набора без исходного словаря	1322	860

D.2.2 Размер документа Быстрого информационного набора со ссылкой на внешний словарь является наименьшим, также как и его размер после сжатия методом GZIP. Отношение размера сжатого методом GZIP документа к размеру документа Быстрого информационного набора указывает, что документ Быстрого информационного набора содержит малое количество избыточной информации.

D.2.3 Во всех случаях размер документов Быстрого информационного набора, сжатых методом GZIP, меньше размера документов XML 1.0, сжатых методом GZIP. Более того, размер без сжатия документа Быстрого информационного набора со ссылкой на внешний словарь меньше, чем размер документа XML 1.0, сжатого методом GZIP.

D.3 Пример заказа UBL

D.3.1 Пример заказа столярных изделий

D.3.1.1 Пример заказа UBL взят из [1]. Именно Заказ столярных изделий был выбран (см. xml/joinery/UBL-Order-1.0-Joinery-Example.xml) по следующим причинам:

a) он является примером из реального мира и был разработан независимо от настоящей Рекомендации | Международного стандарта без какой-либо подстройки под технологию Быстрого информационного набора;

b) он находится в свободном доступе;

c) в нем активно используются пространства имен XML, и таким образом, он является хорошим примером поддержки Быстрым информационным набором пространств имен XML.

D.3.1.2 В Заказ столярных изделий были внесены следующие изменения:

a) последние три элемента **OrderLine** были удалены;

ПРИМЕЧАНИЕ 1. – Это позволяет уменьшить размер документа XML 1.0 до разумных размеров для целей использования в рамках настоящей Рекомендации | Международного стандарта.

b) все пробельные символы были удалены.

ПРИМЕЧАНИЕ 2. – Это представляет собой более реалистичный случай использования Информационных наборов XML, которые могут быть сериализованы, переданы по сети и проанализированы.

D.3.2 Документ XML 1.0 Заказа столярных изделий

Документ XML 1.0 Заказа столярных изделий с изменениями, обозначенными в п. D.3.1.2 a, но с сохранением пробельных символов для удобства чтения, представлен ниже:

```
<?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="http://www.w3.org/2001/XMLSchema-instance"
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>
                                <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 информационного набора для заказа UBL с внешним словарем

Внешний словарь документа Быстрого информационного набора представлен в п. D.4.1. Октеты (как шестнадцатеричные символы) представлены в п. D.4.2. Детальные объяснения некоторых последовательностей октетов приведены в п. D.4.2 и D.4.3. Документ Быстрого информационного набора не может рассматриваться как самоописывающийся, поскольку для воспроизведения полного Информационного набора XML требуется внешняя информация (внешний словарь).

ПРИМЕЧАНИЕ. – Документ Быстрого информационного набора может быть обработан анализатором, который не может получить словарные таблицы по данному URI, однако ссылки на индексы словарных таблиц не могут быть разыменованы (поставлены в соответствие с записями словарных таблиц) для получения информации, необходимой для генерации свойств единиц информации.

D.4.1 Внешний словарь заказа UBL

D.4.1.1 Внешний словарь документа Быстрого информационного набора определен как окончательный словарь, полученный из Информационного набора XML примера заказа UBL (см. п. D.3.1.2) и впоследствии измененный с тем, чтобы:

a) не содержать единиц информации **character**;

b) содержать пустые свойства **[normalized value]** единиц информации **attribute**.

ПРИМЕЧАНИЕ 1. – Это представляет реалистичный сценарий, где заранее не известно, каким будет определяемое приложением содержание Информационного набора XML (единицы информации **character** и или свойства **[normalized value]** единиц информации **attribute**).

ПРИМЕЧАНИЕ 2. – Не предполагается, что документ, подлежащий сериализации, будет использоваться для создания внешнего словаря. Ожидается, что инструменты используют схему и, возможно, экземпляры Информационного набора XML схемы для частотного анализа строк и уточненных имен таким образом, что меньшие значения индексов будут назначены более часто используемой информации (например, частота появления свойств **[local name]** в Информационных наборах XML может представлять собой степенной ряд).

D.4.1.2 URI для внешнего словаря: **urn:oasis:names:tc:ubl:Order:1.0:joinery:example**.

D.4.1.3 В Таблице D.2 представлен словарь Информационного набора XML заказа UBL (словарные таблицы). В столбце 1 перечислены индексы словарных таблиц (Индекс), в столбце 2 перечислены записи словарной таблицы для таблицы PREFIX (Запись "префикс"), в столбце 3 перечислены записи словарной таблицы для таблицы NAMESPACE NAME (Запись "имя пространства имен"), в столбце 4 перечислены записи словарной таблицы для таблицы LOCAL NAME (Запись "локальное имя"), в столбце 5 перечислены записи словарной таблицы для таблицы ELEMENT NAME (Запись "имя элемента"), в столбце 6 перечислены записи словарной таблицы для таблицы ATTRIBUTE NAME (Запись "имя атрибута"). Значения индексов для заменителей 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 "_" определяет, что значение отсутствует (такая ситуация возникает только для значений компонентов **`prefix-name-string-index`** и **`namespace-name-string-index`**).

ПРИМЕЧАНИЕ 1. – Первая запись (индекс `1`), для префикса и имени пространства имен, соответствующих префиксу XML "xml" и имени пространства имен XML "http://www.w3.org/XML/1998/namespace", является встроенной (см. п. 7.2.21 и п. 7.2.22).

ПРИМЕЧАНИЕ 2. – Длинные записи имен пространства имен (URI) были урезаны.

ПРИМЕЧАНИЕ 3 – Для первой записи "имя элемента" (индекс `1`) нет ссылки на префикс (поскольку значение отсутствует и представлено как "_"), есть ссылка на седьмую запись "имя пространства имен" (индекс `7`) для свойства **[namespace name]** ("urn:oasis:names:tc:ubl:Order:1:0"), и есть ссылка на первую запись "локальное имя" (индекс `1`) для свойства **[local name]** ("Order").

Таблица D.2 – Словарь для Информационного набора XML заказа UBL

Индекс	Запись "префикс"	Запись "имя пространства имен"	Запись "локальное имя"	Запись "имя элемента"	Запись "имя атрибута"
1	xml	http://www.w3.org/XML/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			RequestedDeliveryDateTime	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 информационного набора (как шестнадцатеричные символы)

В Таблице D.3 представлены октеты документа Быстрого информационного набора для примера заказа UBL, представленного в п. D.3.

ПРИМЕЧАНИЕ. – Шестнадцатеричные символы, содержащие биты, соответствующие идентификации и указателю конца единиц информации, выделены подчеркиванием.

Таблица D.3 – Октеты документа Быстрого информационного набора (как шестнадцатеричные символы)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e00100002010002f75726e3a6f617369733a6e616d65733a74633a75626c3a4f
000020	726465723a313a303a6a6f696e6572793a6578616d706c6578cf8181cf8282cf
000040	8383cf8484cf8585cd86f00000083b75726e3a6f617369733a6e616d65733a74
000060	633a75626c3a4f726465723a313a30202e2e2f2e2e2f7873642f6d61696e646f
000080	632f55424c2d4f726465722d312e302e787364f00182075330332d3033343235
0000a0	37f00282073230303332d30322d3033f003040506820e4a65727279204275696c
0000c0	64657220706c63ff070882074d61727368204c616e65f00982044e6f77686572
0000e0	65f00a82054e52313820345858f00b82044e6f72666f6c6bff0c068206457661
000100	20427269636bffff0d040506821353706563696 16c6973742057696e646f7773
000120	20706c63ff070e820b536e6f7768696c6c20576f726b73f009820b4c6974746c
000140	6520536e6f72696e67f00a8204534d3220334e57f00b820757686572657365736869
000160	7265ffff0f108210323030332d30322d32345430303a30303a3030f01108820a
000180	52697665727369646520526 42ef00e8217506c6f742031372c20576869746577
0001a0	617465722045737461746 5f00982065768657473746f6e65f00b82064d696464
0001c0	6c65736578fff01213149041f0550143756e6974f09032f01617189202323336
0001e0	5756f0191a9201776f6f64f01b9201736f6674ff191a820366696e697368f01b
000200	82037072696d6564ff191a8205666974746 96e6773f01b9202736174696eff19
000220	1a8204676c617a696e67f01b820373696e676c65ffffff1213149042f0550180
000240	f09033f016171892023334305457f0191a920168616e64f01b915248ff191aa3
000260	f01b92016861726 4ff191a820366696e697368f01b9202737461696eff191a82
000280	0566697474696e6773f01b92026272617373ff191a8204676c617a696e67f01b
0002a0	8203646f75626c65ffffffff
0002ac	

D.4.3 Объяснение кодирования

D.4.3.1 Кодирование единицы информации document и единицы информации Order element

Ниже приводится подробное объяснение исходного кодирования документа Быстрого информационного набора (включая URI внешнего словаря) и корневой единицы информации element. В частности, объясняется кодирование единицы информации **document**, последовательности единиц информации **namespace**, единицы информации **element** и единицы информации **attribute**. В Таблице D.4 представлен фрагмент документа Быстрого информационного набора для кодирования единицы информации **document** из единицы информации **Order element** из п. D.3.2. В таблице D.5 данное кодирование подробно описывается. Этот фрагмент в формате 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="http://www.w3.org/2001/XMLSchema-instance"
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.4 – Октеты фрагмента (как шестнадцатеричные символы)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e00100002010002f75726e3a6f617369733a6e616d65733a74633a75626c3a4f
000020	726465723a313a303a6a6f696e6572793a6578616d706c6578cf8181cf8282cf
000040	8383cf8484cf8585cd86f00000083b75726e3a6f617369733a6e616d65733a74
000060	633a75626c3a4f726465723a313a30202e2e2f2e2e2f7873642f6d61696e646f
000080	632f55424c2d4f726465722d312e302e787364f0

Таблица D.5 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
00 01	11100000 (e0) 00000000 (00)	Эти октеты присутствуют в начале каждого документа Быстрого информационного набора (см. п. 12.6).	единица информации **document**
02 03	00000000 (00) 00000001 (01)	Эти октеты представляют собой кодирование номера версии (см. п. 12.9).	
04 05 06	00100000 (20) 00010000 (10) 00000000 (00)	Эти октеты представляют собой кодирование наличия исходного словаря и ссылки на внешний словарь из исходного словаря. В первом бите октета, стоящего на позиции 04_{16}, со значением 20_{16}, содержится значение '0' (дополнение) (см. п. 12.8). В третьем бите содержится значение '1', обозначая, что компонент **initial-vocabulary** присутствует, а остальные шесть необязательных компонентов отсутствуют (см. п. C.2.3). В первых трех битах октета, стоящего на позиции 05_{16}, со значением 10_{16}, содержатся три значения '0' (дополнение) (см.п. C.2.5). В четвертом бите содержится значение '1', обозначая, что компонент **external-vocabulary** из **initial-vocabulary** присутствует. В последних четырех битах содержатся значения '0' (с пятого по восьмой биты), обозначая, что четыре из двенадцати других необязательных компонентов отсутствуют (см. п. C.2.5.1). Во всех битах октета, стоящего на позиции 06_{16}, со значением 00_{16}, содержатся значения '0', обозначая, что последние восемь из двенадцати необязательных компонентов отсутствуют (см. п. C.2.5.1).	
07 08 37	00101111 (2f) 01110101 (75) 01100101 (65)	Эти октеты представляют собой кодирование URI внешнего словаря. В первом бите октета, стоящего на позиции 07_{16}, со значением $2f_{16}$, содержится значение '0' (дополнение) (см. п. C.2.5.2). URI кодируется как символы UTF-8 (см. п. C.22). Во втором бите содержится значение '0', обозначая, что длина URI больше либо равна 1_{10} октету и меньше либо равна 64_{10} октетам и длина, за вычетом нижней границы, кодируется в битах от третьего до восьмого как целое число без знака (см. п. C.22.3.1). Целое число без знака равно 47_{10} и длина равна 48_{10} (нижняя граница равна 1_{10}). 48_{10} октетов закодированных символов UTF-8 (URI) кодируются начиная с октета, стоящего на позиции 08_{16} и заканчивая октетом, стоящим на позиции 37_{16}.	
38	01111000 (78)	Данный октет представляет собой исходное кодирование дочернего элемента единицы информации **document**. В первом бите октета, стоящего на позиции 38_{16}, со значением 78_{16}, содержится значение '0' (идентификация), обозначая, что у единица информации **document** есть дочерний элемент, и этот дочерний элемент является единицей информации **element** (см. п. C.2.11.2). Во втором бите содержится значение '1', обозначая, что у данной единицы информации **element** есть атрибуты (см. п. C.3.3). В битах с третьего по шестой содержатся значения '1110' за которыми следуют значения '00' (дополнение), содержащиеся в битах с седьмого по восьмой, обозначая, что присутствуют единицы информации **attribute** пространства имен (см. п. C.3.4.1).	единица информации **element** со свойством **[namespace attribute]**

Таблица D.5 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
39 3a 3b	11001111 (cf) 10000001 (81) 10000001 (81)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с индексированными свойствами **[prefix]** и **[normalized value]**. В битах с первого по шестой (с первого по пятый) октета, стоящего на позиции 39_{16}, со значением cf_{16}, содержатся значения '110011' (идентификация), обозначая, что единица информации **attribute** пространства имен присутствует (см. п. C.3.4.2). В седьмом бите содержится значение '1', обозначая, что свойство **[prefix]** присутствует. В восьмом бите содержится значение '1', обозначая, что присутствует свойство **[normalized value]**. В первом бите октета, стоящего на позиции $3a_{16}$, со значением 81_{16}, содержится значение '1', обозначая, что индекс закодирован, и этот индекс идентифицирует свойство **[prefix]** в таблице PREFIX (см. п. C.13.4). Во втором бите содержится значение '0', обозначая, что индекс больше либо равен 1_{10} и меньше либо равен 64_{10}, и индекс кодируется в битах с третьего по восьмой как целое число без знака(см. п. C.25.2). Целое число без знака равно 1_{10} и индекс равен 2_{10} (нижняя граница равна 1_{10}), что в результате дает значение свойства **[prefix]** "res" после разыменования индекса по таблице PREFIX. В первом бите октета, стоящего на позиции $3b_{16}$, со значением 81_{16}, содержится значение '1', обозначающее, что индекс закодирован, и этот индекс идентифицирует свойство **[normalized value]** в таблице NAMESPACE NAME (см. п. C.13.4). Во втором бите содержится значение '0', обозначая, что индекс больше либо равен 1_{10} и меньше либо равен 64_{10}, и индекс кодируется в битах с третьего по восьмой как целое число без знака (см. п. C.25.2). Целое число без знака равно 1_{10} и индекс равен 2_{10} (нижняя граница равна 1_{10}), что в результате дает значение свойства **[normalized value]** равное ".ResponseCode:1.0" после разыменования по таблице NAMESPACE NAME.	**xmlns:res= "….ResponseCode:1.0"**
3c 3d 3e	11001111 (cf) 10000010 (82) 10000010 (82)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с индексированными свойствами **[prefix]** и **[normalized value]**. Индекс свойства **[prefix]** равен 3_{10}, что после разыменования по таблице PREFIX дает значение "cbc". Индекс свойства **[normalized value]** равен 3_{10}, что после разыменования по таблице NAMESPACE NAME дает значение "….sicComponents:1.0".	**xmlns:cbc= "….sicComponents:1.0"**
3f 40 41	11001111 (cf) 10000011 (83) 10000011 (83)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с индексированными свойствами **[prefix]** и **[normalized value]**.	**xmlns:cac= "….ateComponents:1.0"**
42 43 44	11001111 (cf) 10000100 (84) 10000100 (84)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с индексированными свойствами **[prefix]** и **[normalized value]**.	**xmlns:cur= "….CurrencyCode:1.0"**
45 46 47	11001111 (cf) 10000101 (85) 10000101 (85)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с индексированными свойствами **[prefix]** и **[normalized value]**.	**xmlns:xsi= "….Schema-instance"**
48 49	11001101 (cd) 10000110 (86)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с индексированными свойством **[normalized value]**. В седьмом бите октета, стоящего на позиции 48_{16}, со значением cd_{16}, содержится значение '0', обозначая, что свойство **[prefix]** отсутствует, а в восьмом бите содержится значение '1', обозначая, что свойство **[normalized value]** присутствует.	**xmlns="….Order:1:0"**
4a	11110000 (f0)	Этот октет представляет собой кодирование указателя конца для последовательности единиц информации **attribute**. В первых четырех битах октета, стоящего на позиции $4a_{16}$, со значением $f0_{16}$, содержатся значения '1111' (указатель конца) (биты с первого по четвертый) и эти биты являются указателем конца для последовательности. Четыре из шести значений '0' (дополнение) содержатся в битах с пятого по восьмой (см. п. C.3.4.3).	

Таблица D.5 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
4b	00000000 (00)	Этот октет представляет собой кодирование индексированного уточненного имени единицы информации **element**. В первом и втором битах октета, стоящего на позиции $4b_{16}$, со значением 00_{16}, содержатся два оставшихся из шести значений '0' (дополнение) (см. п. C.3.4.3). В третьем бите содержится значение '0', обозначая, что уточненное имя не является буквенным уточненным именем (см. п. C.18.3) и является индексированным. Индекс больше либо равен 1_{10} и меньше либо равен 32_{10}, и индекс кодируется в битах с четвертого по восьмой как целое число без знака (см .п. C.27.2). Целое число без знака равно 0_{10} и индекс равен 1_{10} (нижняя граница равна 1_{10}), что в результате дает уточненное имя со значением свойства **[namespace name]** "….Order:1.0" и значением свойства **[local name]** "Order" (для этого уточненного имени нет свойства **[prefix]**) после разыменования по таблице ELEMENT NAME.	**<Order ….**
4c 4d 4e 4f 92	00000000 (00) 00001000 (08) 01111011 (3b) 01110101 (75) 01101000 (64)	Эти октеты представляют собой кодирование единицы информации **attribute** с индексированным уточненным именем и свойства **[normalized value]**. Присутствие единиц информации **attribute** было обозначено в октете, стоящем на позиции 38_{16} (второй бит имеет значение '1'). В первом бите октета, стоящего на позиции $4c_{16}$, со значением 00_{16}, содержится значение '0' (идентификация), обозначая, что единица информации **attribute** присутствует (см. п. C.3.6.1). Во втором бите содержится значение '0', обозначая, что уточненное имя не является буквенным (см. п. C.17.3) и является индексированным. Индекс больше либо равен 1_{10} и меньше либо равен 64_{10}, и этот индекс кодируется в битах с третьего по восьмой как целое число без знака (см. п. C.25.2). Целое число без знака равно 0_{10} и индекс равен 1_{10} (нижняя граница равна 1_{10}), что в результате дает уточненное имя со значением свойства **[prefix]** "xsi", значением свойства **[namespace name]** "….Schema-instance" и значением свойства **[local name]** "schemaLocation" после разыменования по таблице ATTRIBUTE NAME. Октет, стоящий на позиции $4d_{16}$, со значением 08_{16}, представляет собой исходное кодирование "не идентифицирующей" строки или индекса (см. п. C.14) для свойства **[normalized value]**. В первом бите содержится значение '0', обозначая, что присутствует строка буквенных символов (см. п. C.14.3). Во втором бите содержится значение '0', обозначая, что строку буквенных символов не следует добавлять в таблицу ATTRIBUTE VALUE. В третьем и четвертом битах содержатся значения '0', обозначая, что формат кодирования строки - UTF-8 (см. п. C.19.3.1). В пятом и шестом битах содержатся, соответственно, значения '1' и '0', обозначая, что длина октетов закодированных символов UTF-8 (свойство **[normalized value]**) больше либо равна 9_{10} октетам и меньше либо равна 264_{10} октетам, и длина, за вычетом нижней границы, кодируется в восьми битах следующего октета как целое число без знака (см. п. C.23.3.2). В седьмом и восьмом битах содержатся значения '0' (дополнение) (см. п. C.23.3.2). Октет, стоящий на позиции $4e_{16}$, со значением $3b_{16}$, представляет собой кодирование целого числа без знака. Длина октетов закодированных символов UTF-8 равна 68_{10} (нижняя граница равна 9_{10}). 68_{10} октетов закодированных символов UTF-8 (свойства **[normalized value]**) кодируются с октета, стоящего на позиции $4f_{16}$ до октета, стоящего на позиции 92_{16}.	**xsi:schemaLocation="…."**
93	11110000 (f0)	Этот октет представляет собой кодирование указателя конца последовательности единиц информации **attribute**. В первых четырех битах октета, стоящего на позиции 93_{16}, со значением $f0_{16}$, содержатся значения '1111' (с первого по четвертый бит) и эти биты являются указателем конца последовательности. Четыре значения '0' (дополнения) присутствуют (с пятого по восьмой бит), поскольку у единицы информации **Order** типа **element** есть дочерние элементы (см. п. D.3.2).	

D.4.3.2 Кодирование единицы информации **Address** типа **element** из единицы информации **BuyerParty** типа **element**

Ниже приводится детальное объяснение кодирования единицы информации **Address** типа **element** из единицы информации **BuyerParty** типа **element** документа Быстрого информационного набора. В частности, объясняется кодирование единиц информации **element** и единиц информации **character**. В Таблице D.6 представлен фрагмент документа Быстрого информационного набора для кодирования единицы информации **Address** типа **element** из единицы информации **BuyerParty** типа **element** из п. D.3.2. В Таблице D.7 приводится подробное описание кодирования. Этот фрагмент в формате 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 (как шестнадцатеричные числа)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
0000c0	070882074d61727368204c616e65f00982044e6f77686572
0000e0	65f00a82054e52313820345858f00b82044e6f72666f6c6bff

Таблица D.7 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
c8	00000111 (07)	Этот октет представляет собой кодирование единицы информации **Address** типа **element**. В первом бите октета, стоящего на позиции $c8_{16}$, со значением 07_{16}, содержится значение '0' (идентификация), обозначая, что у единицы информации **element** есть дочерний элемент (дочерний элемент единицы информации **Party** типа **element**), и этот дочерний элемент является единице информации **element** (см. п. C.3.7.2). Во втором бите содержится значение '0', обозначая, что у единицы информации **element** нет атрибутов (см. .п. C.3.3). В третьем бите содержится значение '0', обозначая, что уточненное имя не является буквенным (см. п. C.18.3) и проиндексировано. Индекс больше либо равен 1_{10} и меньше либо равен 32_{10}, и этот индекс кодируется в битах с четвертого по восьмой как целое число без знака (см. п. C.27.2). Целое число без знака равно 7_{10} и индекс равен 8_{10} (нижняя граница равна 1_{10}), что в результате дает уточненное имя со значением **[prefix]** "cac", значением свойства **[namespace name]** "….gateComponents:1.0" и значением свойства **[local name]** "Address" после разыменования по таблице ELEMENT NAME.	**<cac:Address>**
c9	00001000 (08)	Этот октет представляет собой кодирование единицы информации **StreetName** типа **element**. Единица информации **element** имеет индекс 9_{10}, что в результате дает уточненное имя со значением **[prefix]** "cbc", значением свойства **[namespace name]** "….BasicComponents:1:0" и значением свойства **[local name]** "StreetName" после разыменования по таблице ELEMENT NAME.	**<cbc:StreetName>**

Таблица D.7 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
ca cb cc d5	10000010 (82) 00000111 (07) 01001101 (4d) 01100101 (65)	Эти октеты представляют собой кодирование единицы информации **character** из единицы информации **StreetName** типа **element**. В первых двух битах октета (с первого по второй биты), стоящего на позиции ca_{16}, со значением 82_{16}, содержится значением '10' (идентификация), обозначая, что у единицы информации **element** есть дочерний элемент (дочерний элемент единицы информации **StreetName** типа **element**), и этот дочерний элемент представляет собой блок единиц информации **character** (см.п. С.3.7.5). В третьем бите содержится значение '0', обозначая, что присутствует строка буквенных символов (см. п. С.15.3). В четвертом бите содержится значение '0', обозначая, что данную строку буквенных символов не следует добавлять к таблицу CONTENT CHARACTER CHUNK. Как в пятом, так и в шестом битах содержатся значения '0', обозначая, что формат кодирования блока – UTF-8 (см. п. С.20.3.1). В седьмом и восьмом битах содержатся значения '1' и '0' соответственно, обозначая, что длина октетов закодированных символов UTF-8 (блок единиц информации **character**) больше либо равна 3_{10} октетам и меньше либо равна 258_{10} октетам, и что длина, за вычетом нижней границы, кодируется в восьми битах последующего октета как целое число без знака (см. п.С.24.3.2). Октет, стоящий на позиции cb_{16}, со значением 07_{16}, представляет собой целое число без знака. Длина октетов закодированных символов UTF-8 равна 10_{10} (нижняя граница равна 3_{10}). 10_{10} октетов закодированных символов UTF-8 кодируются начиная с октета, стоящего на позиции cc_{10} и заканчивая октетом, стоящи на позиции $d5_{10}$.	единицы информации **character** "Marsh Lane"
d6	11110000 (f0)	Этот октет является указателем конца для единицы информации **StreetName** типа **element**. В первых четырех битах октета (биты с первого по четвертый) $d6_{16}$, со значением $f0_{16}$, содержатся значения '1111' (указатель конца) и эти биты являются указателем конца для единицы информации **StreetName** типа **element** (см.п. С.3.8). В битах с пятого по восьмой содержатся значения '0' (дополнение) поскольку имеет место последующий дочерний элемент (элемент того же уровня) (единица информации **CityName** типа **element**) (см.п. С.3.7.1).	**</cbc:StreetName>**
d7	00001001 (09)	Этот октет является кодированием единицы информации **CityName** типа **element**. Данная единица информации **element** имеет индекс 10_{10}, что в результате дает уточненное имя со значением **[prefix]** "cbc", значением свойства **[namespace name]** "….BasicComponents:1:0" и значением свойства **[local name]** "CityName" после разыменования по таблице ELEMENT NAME.	**<cbc:CityName>**
d8 d9 da e0	10000010 (82) 00000100 (04) 01001110 (4e) 01100101 (65)	Эти октеты представляют собой кодирование единиц информации **character** из единицы информации **CityName** типа **element**. 7_{10} октетов закодированных символов UTF-8 кодируются начиная с октета, стоящего на позиции da_{16} и заканчивая октетом, стоящим на позиции $e0_{16}$.	единицы информации **character** "Nowhere"
e1	11110000 (f0)	Этот октет является указателем концы для единицы информации **CityName** типа **element**.	**</cbc:CityName>**
e2	00001010 (0a)	Этот октет представляет собой кодирование единицы информации **PostalZone** типа **element**. Данная единица информации **element** имеет индекс 11_{10}, что в результате дает уточненное имя со значением **[prefix]** "cbc", значением свойства **[namespace name]** "….BasicComponents:1:0" и значением свойства **[local name]** "PostalZone" после разыменования по таблице ELEMENT NAME.	**<cbc:PostalZone>**
e3 e4 e5 .... ec	10000010 (82) 00000101 (05) 01001110 (4e) 01011000 (58)	Эти октеты представляют собой кодирование единиц информации **character** из единицы информации **PostalZone** типа **element**. 8_{10} октетов закодированных символов UTF-8 кодируются, начиная с октета, стоящего на позиции $e5_{16}$ и заканчивая октетом, стоящим на позиции ec_{16}.	единицы информации **character** "NR18 4XX"

Таблица D.7 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
ed	11110000 (f0)	Этот октет является указателем конца для единицы информации **PostalZone** типа **element**.	**</cbc:PostalZone>**
ee	00001011 (0b)	Этот октет является кодированием единицы информации **CountrySubentity** типа **element**. Данная единица информации **element** имеет индекс 12_{10}, что в результате дает уточненное имя со значением **[prefix]** "cbc", значением свойства **[namespace name]** "….BasicComponents:1:0" и значением свойства **[local name]** "CountrySubentity" после разыменования по таблице ELEMENT NAME.	**<cbc:CountrySubentity>**
ef f0 f1 .... f7	10000010 (82) 00000100 (04) 01001110 (4e) 01101011 (6b)	Эти октеты представляют собой кодирование единиц информации **character** из единицы информации **CountrySubentity** типа **element**. 7_{10} октетов закодированных символов UTF-8 кодируются, начиная с октета, стоящего на позиции $f1_{16}$ и заканчивая октетом, стоящим на позиции $f7_{16}$.	единицы информации **character** "Norfolk"
f8	11111111 (ff)	Этот октет является указателем конца для единицы информации **CountrySubentity** типа **element** и для единицы информации **Address** типа **element**. В первых четырех битах октета (биты с первого по четвертый), стоящего на позиции $f8_{16}$,со значением ff_{16}, содержатся значения '1111' (указатель конца) и эти биты являются указателем конца для единицы информации **CountrySubentity** типа **element** (см.п. C.3.8). Последние четыре бита (биты с пятого по восьмой) содержат значения '1111' и являются указателем конца для единицы информации **Address** типа **element** (см.п. C.3.8).	**</cbc:CountrySubentity>** **</cac:Address>**

D.5 Документ Быстрого информационного набора заказа UBL без исходного словаря

В п. D.5.1 представлены октеты документа Быстрого информационного набора (как шестнадцатеричные символы). Подробные объяснения некоторых последовательностей октетов из п. D.5.1 приведены в п. D.5.2. Окончательный словарь этого документа Быстрого информационного набора и старого документа Быстрого информационного набора будут идентичны, поскольку индексы словарных таблиц во внешнем словаре генерируются в том же порядке. Поскольку строки встроены в документ Быстрого информационного набора, размер его будет больше. Увеличение составит 635_{10} байт (размер документа Быстрого информационного набора минус размер старого документа Быстрого информационного набора), что составляет примерно половину размера документа (для документов большего размера эта разница будет меньше, поскольку существует тенденция к независимости размера словаря от размера документа). В отличие от старого документа Быстрого информационного набора, этот документ может быть признан самоописывающимся, поскольку Информационный набор XML может быть воспроизведен без использования какой-либо внешней информации (внешнего словаря).

D.5.1 Октеты документа Быстрого информационного набора (как шестнадцатеричные символы)

В Таблице D.8 представлены октеты документа Быстрого информационного набора для примера заказа UBL, представленного в п. D.3.

ПРИМЕЧАНИЕ. – Шестнадцатеричные символы, содержащие биты, соответствующие идентификации и указателю конца единиц информации, выделены подчеркиванием.

Таблица D.8 – Октеты документа Быстрого информационного набора (как шестнадцатеричные символы)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e001000000078cf027265733e75726e3a6f617369733a6e616d65733a74633a75
000020	626c3a636f64656c6973743a41636b6e6f776c656467656d656e74526573706f
000040	6e7365436f64653a313a30cf026362632f75726e3a6f617369733a6e616d6573
000060	3a74633a75626c3a436f6d6d6f6e4261736963436f6d706f6e656e74733a313a
000080	30cf026361633375726e3a6f617369733a6e616d65733a74633a75626c3a436f
0000a0	6d6d6f6e41676772656761746 5436f6d706f6e656e74733a313a30cf02637572
0000c0	2f75726e3a6f617369733a6e616d65733a74633a75626c3a636f64656c697374
0000e0	3a43757272656e6379436f64653a313a30cf027873692868747 4703a2f2f7777
000100	772e77332e6f72672f323030312f584d4c536368656d612d696e7374616e6365
000120	cd1f75726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a31
000140	3a30f03d86044f726465727b85850d736368656d614c6f636174696f6e083b75
000160	726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a313a3020
000180	2e2e2f2e2e2f7873642f6d61696e646f632f55424c2d4f726465722d312e302e
0001a0	787364f03d860742757965727349448207533033 2d303334323537f03f828208
0001c0	4973737565446174658207323030332d30322d3033f03f838309427579657250
0001e0	617274793f838304506172747 93f83830850617274794e616d653f8282034e61
000200	6d65820e4a65727279204275696c64657220706c63ff3f838306416464726573
000220	733f828209537472656574e616d6582074d61727368204c616e65f03f828207
000240	436974794e616d6582044e6f7768657265f03f828209506f7374616c5a6f6e65
000260	82054e52313820345858f03f82820f436f756e747279537562656e7469747982
000280	044e6f72666f6c6bff3f838306436f6e7461637406820645766120427269636b
0002a0	ffff3f83830a53656c6c65725061727479040506821353706563696 16c697374
0002c0	2057696e646f777320706c63ff073f82820b4275696c64696e674e616d65820b
0002e0	536e6f7768696c6c20576f726b73f009820b4c6974746c6520536e6f72696e67
000300	f00a8204534d3220334e57f00b8207576865726573689726 5ffff3f83830744
000320	656c69766572793f828218526571756573746564446 56c697665727944617465
000340	54696d6582103230303 0332d30322d32345430303a30303a3030f03f83830e4465
000360	6c697665727941646472657373 08820a52697665727369646520526 42ef00e82
000380	17506c6f742031372c20576869746577617465722045737461746 5f009820657
0003a0	68657473746f6e65f00b82064d6964646c65736578fff03f8383084f72646572
0003c0	4c696e653f8383074c696e654974656d3f8383829041f07f8282075175616e74
0003e0	69747978 0f7175616e74697479556e6974436f646543756e6974f09032f03f83
000400	83034974656d3f83831853656c6c6572734974656d4964656e74696669636174
000420	696f6e3f838301494492023233365756f03f838310506879736963616c417474
000440	7269627574653f83830a417474726962757465494492 01776f6f64f03f82820a
000460	446573637269707 4696f6e9201736f6674ff191a820366696e697368f01b8203
000480	7072696d6564ff191a820566697474696e6773f01b9202736174696eff191a82
0004a0	04676c617a696e67f01b820373696e676c65ffffff1213149042f0550180f090
0004c0	33f016171892023334305457f0191a920168616e64f01b915248ff191aa3f01b
0004e0	920168617264ff191a820366696e697368f01b9202737461696eff191a820566
000500	697474696e6773f01b9202672617373ff191a8204676c617a696e67f01b8203
000520	646f75626c65ffffffff
00052a	

D.5.2 Объяснение кодирования

D.5.2.1 Кодирование единицы информации document и единицы информации Order типа element

Ниже приводится подробное объяснение исходного кодирования документа Быстрого информационного набора и корневой единицы информации element. В частности, объясняется кодирование единицы информации **document**, последовательности единиц информации **namespace**, единицы информации **element** и единицы информации **attribute**. В Таблице D.9 представлен фрагмент документа Быстрого информационного набора для кодирования единицы

информации **document** из единицы информации **Order element** из п. D.3.2. В таблице D.10 данное кодирование подробно описывается. Этот фрагмент в формате 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="http://www.w3.org/2001/XMLSchema-instance"
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 (как шестнадцатеричные символы)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000000	e00100000078cf027265733e75726e3a6f617369733a6e616d65733a74633a75
000020	626c3a636f64656c6973743a41636b6e6f776c656467656d656e74526573706f
000040	6e7365436f64653a313a30cf026362632f75726e3a6f617369733a6e616d6573
000060	3a74633a75626c3a436f6d6d6f6e4261736963436f6d706f6e656e74733a313a
000080	30cf026361633375726e3a6f617369733a6e616d65733a74633a75626c3a436f
0000a0	6d6d6f6e41676772656761746543 6f6d706f6e656e74733a313a30cf02637572
0000c0	2f75726e3a6f617369733a6e616d65733a74633a75626c3a636f64656c697374
0000e0	3a43757272656e6379436f64653a313a30cf027873692868747 4703a2f2f7777
000100	772e77332e6f72672f323030312f584d4c536368656d612d696e7374616e6365
000120	cd1f75726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a31
000140	3a30f03d86044f726465727b85850d736368656d614c6f636174696f6e083b75
000160	726e3a6f617369733a6e616d65733a74633a75626c3a4f726465723a313a3020
000180	2e2e2f2e2e2f7873642f6d61696e646f632f55424c2d4f726465722d312e302e
0001a0	787364f0

Таблица D.10 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
00 01	11100000 (e0) 00000000 (00)	Эти октеты присутствуют в начале каждого документа Быстрого информационного набора (см. п. 12.6).	единица информации **document**
02 03	00000000 (00) 00000001 (01)	Эти октеты представляют собой кодирование номера версии (см. п. 12.9).	
04	00000000 (00)	Эти октеты представляют собой кодирование наличия исходного словаря и других документов типа **Document**. В первом бите октета, стоящего на позиции 04_{16}, со значением 00_{16}, содержится значение '0' (дополнение) (см.п. 12.8). В битах со второго по восьмой содержатся значения '0000000', обозначая, что все необязательные компоненты типа **Document** отсутствуют (включая компонент **initial-vocabulary**, отсутствие которого обозначено третьим битом, см.п. C.2.3).	
05	01111000 (78)	Этот октет представляет собой исходное кодирование единицы информации **document**. В первом бите октета, стоящего на позиции 05_{16}, со значением 78_{16}, содержится значение '0' (идентификация), обозначая, что у единицы информации **document** есть дочерний элемент, и этот дочерний элемент является единицей информации **element** (см. п. C.2.11.2). Во втором бите содержится значение '1', обозначая, что у единицы информации **element** есть атрибуты (см.п. C.3.3). В битах с третьего по шестой содержатся значения '1110', а в битах с седьмого по восьмой содержатся значения '00' (дополнение), обозначая, что присутствуют единицы информации **attribute** (см.п. C.3.4.1).	единица информации **element** со свойством **[namespace attribute]**

Таблица D.10 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный 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 представляют собой кодирование единицы информации **attribute** пространства имен с буквенными свойствами **[prefix]** и **[normalized value]**. В битах с первого по шестой октета, стоящего на позиции 06_{16}, со значением cf_{16}, содержатся значения '110011' (идентификация), обозначая, что присутствует единица информации **attribute** пространства имен (см.п.С.3.4.2). В седьмом бите содержится значение '1', обозначая, что присутствует свойство **[prefix]**. В восьмом бите содержится значение '1', обозначая, что присутствует свойство **[normalized value]**. В первом бите октета, стоящего на позиции 07_{16}, со значением 02_{16}, содержится значение '0', обозначая, что для свойства **[prefix]** закодирована строка буквенных символов(см.п. С.13.3). Во втором бите содержится значение '0', обозначая, что длина закодированных символов UTF-8 больше либо равна 1_{10} и меньше либо равна 64_{10}, и что длина кодируется в битах с третьего по восьмой как целое число без знака (см.п. С.22.3.1). Целое число без знака равно 2_{10} и длина равна 3_{10} (нижняя граница равна 1_{10}). 3_{10} октета закодированных символов UTF-8 (свойства **[prefix]**) кодируются, начиная с октета, стоящего на позиции 08_{16} и заканчивая октетом, стоящим на позиции $0a_{10}$. Строка "res" добавляется в таблицу PREFIX (с индексом равным 2_{10}). В первом бите октета, стоящего на позиции $0b_{16}$, со значением $3e_{16}$, содержится значение '0', обозначая, что для свойства **[normalized value]** закодирована строка буквенных символов (см. п. С.13.3). Во втором бите содержится значение '0', обозначая, что длина закодированных символов UTF-8 больше либо равна 1_{10} и меньше либо равна 64_{10}, и длина кодируется в битах с третьего по восьмой как целое число без знака (см.п. С.22.3.1). Целое число без знака равно 62_{10} и длина равна 63_{10} (нижняя граница равна 1_{10}). 63_{10} октета закодированных символов UTF-8 (свойства **[normalized value]**) кодируются, начиная с октета, стоящего на позиции $0c_{16}$ и заканчивая октетом, стоящим на позиции $4a_{16}$. Строка "….ResponseCode:1:0" добавляется в таблицу NAMESPACE NAME (с индексом равным 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 представляют собой кодирование единицы информации **attribute** пространства имен с буквенными свойствами **[prefix]** и **[normalized value]**. 3_{10} октета закодированных символов UTF-8 (свойства **[prefix]**) кодируются, начиная с октета, стоящего на позиции $4c_{16}$ и заканчивая октетом, стоящим на позиции $4f_{16}$. Строка "cbc" добавляется в таблицу PREFIX (с индексом, равным 3_{10}). 48_{10} октетов закодированных символов UTF-8 (свойства **[normalized value]**) кодируются, начиная с октета, стоящего на позиции 51_{16} и заканчивая октетом, стоящим на позиции 80_{16}. Строка "….sicComponents:1:0" добавляется в таблицу NAMESPACE NAME (с индексом, равным 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 представляют собой кодирование единицы информации **attribute** пространства имен с буквенными свойствами **[prefix]** и **[normalized value]**. 3_{10} октета закодированных символов UTF-8 (свойства **[prefix]**) кодируются, начиная с октета, стоящего на позиции 83_{16} и заканчивая октетом, стоящим на позиции 85_{16}. Строка "cac" добавляется в таблицу PREFIX (с индексом, равным 4_{10}). 52_{10} октета закодированных символов UTF-8 (свойства **[normalized value]**) кодируются, начиная с октета, стоящего на позиции 87_{16} и заканчивая октетом, стоящим на позиции ba_{16}. Строка "….ateComponents:1:0" добавляется в таблицу NAMESPACE NAME (с индексом, равным 4_{10}).	**xmlns:cac=** **"….ateComponents:1:0"**
bb bc bd .... bf	11001111 (cf) 00000010 (02) 01100011 (63) 01100011 (63)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с буквенными свойствами **[prefix]** и **[normalized value]**. 3_{10} октета закодированных символов UTF-8 (свойства **[prefix]**) кодируются, начиная с октета, стоящего на позиции bd_{16} и заканчивая октетом, стоящим на позиции bf_{16}. Строка "cur" добавляется в таблицу PREFIX (с индексом, равным 5_{10}).	**xmlns:cur=** **"….CurrencyCode:1:0"**

Таблица D.10 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
c0 c1 .... f0	00101111 (2f) 01110101 (75) 01110000 (30)	48_{10} октетов закодированных символов UTF-8 (свойства **[normalized value]**) кодируются, начиная с октета, стоящего на позиции $c1_{16}$ и заканчивая октетом, стоящим на позиции $f0_{16}$. Строка "….CurrencyCode:1:0" добавляется в таблицу NAMESPACE NAME (с индексом, равным 5_{10}).	
f1 f2 f3 .... f5 f6 f7 .... 11f	11001111 (cf) 00000010 (02) 01111000 (78) .... 01101001 (69) 00101000 (28) 01101000 (68) 01110111 (77)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с буквенными свойствами **[prefix]** и **[normalized value]**. 3_{10} октета закодированных символов UTF-8 (свойства **[prefix]**) кодируются, начиная с октета, стоящего на позиции $f3_{16}$ и заканчивая октетом, стоящим на позиции $f5_{16}$. Строка "xsi" добавляется в таблицу PREFIX (с индексом, равным 6_{10}). 41_{10} октет закодированных символов UTF-8 (свойства **[normalized value]**) кодируются, начиная с октета, стоящего на позиции $f7_{16}$ и заканчивая октетом, стоящим на позиции $11f_{16}$. Строка "….Schema-instance" добавляется в таблицу NAMESPACE NAME (с индексом, равным 6_{10}).	**xmlns:xsi=** **"….Schema-instance"**
120 121 122 141	11001101 (cd) 00011111 (1f) 01110101 (75) 00110000 (30)	Эти октеты представляют собой кодирование единицы информации **attribute** пространства имен с индексированным свойством **[normalized value]**. 32_{10} октета закодированных символов UTF-8 (свойства **[normalized value]**) кодируются, начиная с октета, стоящего на позиции 122_{16} и заканчивая октетом, стоящим на позиции $141f_{16}$. Строка "….Order:1:0" добавляется в таблицу NAMESPACE NAME (с индексом, равным 7_{10}).	**xmlns="….Order:1:0"**
142	11110000 (f0)	Этот октет является указателем конца для последовательности единиц информации **attribute** пространства имен. В первых четырех битах октета (биты с первого по четвертый), стоящего на позиции 142_{16}, со значением $f0_{16}$, содержатся значения '1111' (указатель конца) и эти биты являются указателем конца для последовательности. Четыре из шести значений '0' (дополнение) содержатся в битах с пятого по восьмой (см.п. C.3.4.3).	
143 144 145 146 .... 14a	00111101 (3d) 10000110 (86) 00000100 (04) 01001111 (4f) 01110010 (72)	Эти октеты представляют собой кодирование буквенного уточненного имени единицы информации **element**. В первом и втором битах октета, стоящего на позиции 143_{16}, со значением $3d_{16}$, содержатся последние два из шести значений '0' (дополнение) (см.п. C.3.4.3). В битах с третьего по пятый содержатся значения '1111', обозначая, что уточненное имя является буквенным (см.п. C.18.3). В седьмом бите содержится значение '0', обозначая, что у уточненного имени нет свойства **[prefix]**. В восьмом бите содержится значение '1', обозначая, что у уточненного имени есть свойство **[namespace name]**. В первом бите октета, стоящего на позиции 144_{16}, со значением 86_{16}, содержится значение '1', обозначая, что свойство **[namespace name]** не является буквенной строкой и является индексированным (см.п. C.13.4). Во втором бите содержится значение '0', обозначая, что индекс больше либо равен 1_{10} и меньше либо равен 64_{10}, и индекс кодируется в битах с третьего по восьмой как целое число без знака (см.п. C.25.2). Целое число без знака равно 6_{10} и индекс равен 7_{10} (нижняя граница равна 1_{10}), что в результате дает свойство **[namespace name]** со значением "….Order:1:0" после разыменования по таблице NAMESPACE NAME. В первом бите октета, стоящего на позиции 145_{16}, со значением 04_{16}, содержится значение '0', обозначая, что для свойства **[local name]** закодирована буквенная строка (см.п. C.13.3). Во втором бите содержится значение '0', обозначая, что длина закодированных символов UTF-8 больше либо равна 1_{10} и меньше либо равна 64_{10}, и длина кодируется в битах с третьего по восьмой как целое число без знака (см.п. C.22.3.1). Целое число без знака равно 4_{10} и длина равна 5_{10} (нижняя гшраница равна 1_{10}). 5_{10} октетов закодированных символов UTF-8 (свойства **[local name]**) кодируются, начиная с бита, стоящего на позиции 146_{16} и заканчивая битом, стоящим на позиции $14a_{16}$. Строка "Order"	**<Order ….**

Таблица D.10 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
		добавляется в таблицу LOCAL NAME (с индексом, равным 1_{10}).	
		Уточненное имя без свойства **[prefix]**, свойство**[namespace name]** со значением "….Order:1:0" (индекс 7_{10}), и свойство **[local name]** со значением "Order" (индекс 1_{10}) добавляются в таблицу ELEMENT NAME (с индексом, равным 1_{10}).	
14b 14c 14d 14e 14f .... 15c 15d 15e 15f 1a2	01111011 (7b) 10000101 (85) 10000101 (85) 00001101 (0d) 01110011 (73) .... 01101110 (6e) 00001000 (08) 00111011 (3b) 01110101 (75) 01100100 (64)	Эти октеты представляют собой кодирование единицы информации **attribute** с буквенным уточненным именем и свойством **[normalized value]**. Наличие единицы информации **attribute** было обозначено в октете, стоящем на позиции 05_{16} (второй бит содержит значение '1'). В первом бите октета, стоящего на позиции $14b_{16}$, со значением $7b_{16}$, содержится значение '0' (идентификация), обозначая, что присутствует единица информации **attribute** (см.п. С.3.6.1). В битах со второго по пятый содержатся значения '1111', обозначая, что уточненное имя является буквенным (см.п. С.17.3). В шестом бите содержится значение '0' (дополнение) (см.п.С.17.3). В седьмом бите содержится значение '1', обозначая, что у уточненного имени есть свойство **[prefix]**. В восьмом бите содержится значение '1', обозначая, что у уточненного имени есть свойство **[namespace name]**. В первом бите октета, стоящего на позиции $14c_{16}$, со значением 85_{16}, содержится значение '1', обозначая, что свойство **[prefix]** не является буквенной строкой и является индексированным (см.п. С.13.4). Во втором бите содержится значение '0', обозначая, что индекс больше либо равен 1_{10} и меньше либо равен 64_{10}, и индекс кодируется в битах с третьего по восьмой как целое число без знака (см.п. С.25.2). Целое число без знака равное 5_{10} и индекс равнее 6_{10} (нижняя граница равнее 1_{10}), что в результате дает значение свойства **[prefix]** "xsi" после разыменования по таблице NAMESPACE NAME. В первом бите октета, стоящего на позиции $14d_{16}$, со значением 85_{16}, содержится значение '1', обозначая, что свойство **[namespace name]** не является буквенной строкой и является индексированным (см.п. С.13.4). Во втором бите содержится значение '0', обозначая, что индекс больше либо равен 1_{10} и меньше либо равен 64_{10}, и индекс кодируется в битах с третьего по восьмой как целое число без знака (см.п. С.25.2). Целое число без знака равное 5_{10} и индекс равнее 6_{10} (нижняя граница равнее 1_{10}), что в результате дает значение свойства **[namespace name]** "….Schema-instance" после разыменования по таблице NAMESPACE NAME. В первом бите октета, стоящего на позиции $14e_{16}$, со значением $0d_{16}$, содержится значение '0', обозначая, что для свойства **[local name]** закодирована строка буквенных символов (см.п. С.13.3). Во втором бите содержится значение '0', обозначая, что длина закодированных символов UTF-8 больше либо равна 1_{10} и меньше либо равна 64_{10}, и длина кодируется в битах с третьего по восьмой как целое число без знака (см.п. С.22.3.1). Целое число без знака равно 13_{10} и длина равна 14_{10} (нижняя граница равна 1_{10}). 14_{10} октетов закодированных символов UTF-8 (свойства **[local name]**) кодируются, начиная с бита, стоящего на позиции $14f_{16}$ и заканчивая битом, стоящим на позиции $15c_{16}$. Строка "SchemaLocation" добавляется в таблицу LOCAL NAME (с индексом, равным 2_{10}). Уточненное имя со значением свойства **[prefix]** "xsi" (с индексом 6_{10}), значением свойства **[namespace name]** "….Schema-instance" (индекс 6_{10}), и значением свойства **[local name]** "schemaLocation" (индекс 2_{10}) добавляется в таблицу ATTRIBUTE NAME (с индексом, равным 1_{10}). Октет, стоящий на позиции $15d_{16}$, со значением 08_{16}, представляет собой исходное кодирование "не идентифицирующей" строки или индекса (см.п. С.14) для свойства **[normalized value]**. В первом бите содержится значение '0', обозначая, что присутствует строка буквенных символов (см.п. С.14.3). Во втором бите содержится значение '0', обозначая, что данную строку буквенных символов не следует	**xsi:schemaLocation="...."**

Таблица D.10 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
		добавлять в таблицу ATTRIBUTE VALUE. Как в третьем, так и в четвертом битах содержатся значения '0', обозначая, что формат кодирования строки - UTF-8 (см.п. C.19.3.1). В пятом и шестом битах содержатся значения '1' и '0' соответственно, обозначая, что длина октетов символов UTF-8 (свойство **[normalized value]**) больше либо равна 9_{10} октетам и меньше либо равна 264_{10} октетам и что длина, за вычетом нижней границы, кодируется в восьми битах последующего октета как целое число без знака (см.п. C.22.3.2). В битах с седьмого по восьмой содержатся значения '0' (дополнение) (см.п. C.22.3.2). Октет, стоящий на позиции $15e_{16}$, со значением $3b_{16}$, представляет собой кодирование целого числа без знака. Длина октетов закодированных символов UTF-8 равна 68_{10} (нижняя граница равна 9_{10}). 68_{10} октетов закодированных символов UTF-8 (свойства **[normalized value]**) кодируются, начиная с октета, стоящего на позиции $15f_{16}$ и заканчивая октетом, стоящим на позиции $1a2_{16}$.	
1a3	11110000 (f0)	Этот октет является указателем конца для последовательности единиц информации **attribute**. В первых четырех битах октета (биты с первого по четвертый), стоящего на позиции $1a3_{16}$, со значением $f0_{16}$, содержатся значения '1111' и эти биты являются указателем конца для последовательности. Присутствуют четыре значения '0' (дополнение) (в битах с пятого по восьмой), поскольку единица информации **Order** типа **element** не имеет дочерних элементов (см.п. D.3.2).	

D.5.2.2 Кодирование единицы информации Address типа element из единицы информации BuyerParty типа element

Ниже приводится детальное объяснение кодирования единицы информации **Address** типа **element** из единицы информации **BuyerParty** типа **element** документа Быстрого информационного набора. В частности, объясняется кодирование единиц информации **element** и единиц информации **character**. В Таблице D.11 представлен фрагмент документа Быстрого информационного набора для кодирования единицы информации **Address** типа **element** из единицы информации **BuyerParty** типа **element** из п. D.3.2. В Таблице D.12 приводится подробное описание кодирования. Этот фрагмент в формате 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 (как шестнадцатеричные символы)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
000200	3f838306416464726573
000220	733f828209537472656574e616d6582074d61727368204c616e65f03f828207
000240	436974794e616d6582044e6f7768657265f03f828209506f7374616c5a6f6e65
000260	82054e52313820345858f03f82820f436f756e747279537562656e7469747982
000280	044e6f72666f6c6bff

Таблица D.12 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
216 217 218 219 21a 220	00111111 (3f) 10000011 (83) 10000011 (83) 00000110 (06) 01000001 (41) 01110011 (73)	Эти октеты представляют собой кодирование единицы информации **Address** типа **element**. В первом бите октета, стоящего на позиции 216_{16}, со значением $3f_{16}$, содержится значение '0' (идентификация) обозначая, что у единицы информации **element** есть дочерний элемент (дочерний элемент единицы информации **Party** типа **element**), и этот дочерний элемент является единицей информации **element** (см.п. С.3.7.2). Во втором бите содержится значение '0', обозначая, что у единицы информации **element** нет атрибутов (см.п. С.3.3). В битах с третьего по пятый содержатся значения '1111', обозначая, что уточненное имя является буквенным (см.п. С.18.3). В седьмом бите содержится значение '1', обозначая, что у уточненного имени есть свойство **[prefix]**. В восьмом бите содержится значение '1', обозначая, что у уточненного имени есть свойство **[namespace name]**. В первом бите октета, стоящего на позиции 217_{16}, со значением 83_{16}, содержится значение '1', обозначая, что свойство **[prefix]** не является буквенной строкой и является индкесным (см.п. С.13.4). Во втором бите содержится значение '0', обозначая, что индекс больше либо равен 1_{10} и меньше либо равен 64_{10}, и индекс кодируется в битах с третьего по восьмой как целое число без знака (см.п. С.25.2). Целое число без знака равно 3_{10} и индекс равен 4_{10} (нижняя граница равна 1_{10}), что в результате дает значение свойства **[prefix]** "cac" после разыменования по таблице PREFIX. В первом бите октета, стоящего на позиции 218_{16}, со значением 83_{16}, содержится значение '1', обозначая, что свойство **[namespace name]** не является буквенной строкой и является индексированным (см.п. С.13.4). Во втором бите содержится значение '0', обозначая, что индекс больше либо равен 1_{10} и меньше либо равен 64_{10}, и индекс кодируется в битах с третьего по восьмой как целое число без знака (см.п. С.25.2). Целое число без знака равно 3_{10} и индекс равен 4_{10} (нижняя граница равна 1_{10}), что в результате дает значение свойства **[namespace name]** "….ateComponents:1:0" после разыменования по таблице NAMESPACE NAME. В первом бите октета, стоящего на позиции 219_{6}, со значением 06, содержится значение '0, обозначая, что для свойства **[local name]** закодирована строка буквенных символов (см.п. С.13.3). Во втором бите содержится значение '0', обозначая, что длина закодированных символов UTF-8 больше либо равна 1_{10} и меньше либо равна 64_{10}, и длина кодируется в битах с третьего во восьмой как целое число без знака (см.п. С.22.3.1). Целое число без знака равно 6_{10} и длина равна 7_{10} (нижняя граница равна 1_{10}). 7_{10} октетов закодированных символов UTF-8 (свойства **[local name]**) кодируются, начиная с бита, стоящего на позиции $21a_{16}$ и заканчивая битом, стоящим на позиции 220_{16}. Строка "Address" добавляется в таблицу LOCAL NAME (с индексом, равным 9_{10}). Уточненное имя со значением свойства **[prefix]** "cac" (индекс 4_{10}), значением свойства **[namespace name]** "……ateComponents:1:0" (индекс 4_{10}), и значением свойства **[local name]** "Order" (индекс 1_{10}) добавляется в таблицу ELEMENT NAME (с индексом, равным 1_{10}).	**<cac:Address>**
221 222 223 224 225 .... 22e	00111111 (3f) 10000010 (82) 10000010 (82) 00001001 (09) 01000001 (53) 01100101 (65)	Эти октеты представляют собой кодирование единицы информации **StreetName** типа **element**. Свойство **[local name]** "StreetName" добавляется в таблицу LOCAL NAME (с индексом, равным 10_{10}). Уточненное имя со значением свойства **[prefix]** "cbc" (индекс 3_{10}), значением свойства **[namespace name]** "…….BasicComponents:1:0" (индекс 3_{10}), и значением свойства **[local name]** "StreetName" (индекс 10_{10}) добавляется в таблицу ELEMENT NAME (с индексом, равным 9_{10})..	**<cbc:StreetName>**

Таблица D.12 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
22f 230 231 .... 23a	10000010 (82) 00000111 (07) 01001101 (4d) 01100101 (65)	Эти октеты представляют собой кодирование единиц информации **character** из единицы информации **StreetName** типа **element**. В первых двух битах октета (биты с первого по второй), стоящего на позиции $22f_{16}$, со значением 82_{16}, содержатся значения '10' (идентификация), обозначая, что у единицы информации **element** есть дочерний элемент (дочерний элемент единицы информации **StreetName** типа **element**), и этот дочерний элемент является блоком единиц информации **character** (см.п. С.3.7.5). В третьем бите содержится значение '0', обозначая, что присутствует строка буквенных символов (см.п. С.15.3). В четвертом бите содержится значение '0', обозначая, что данную строку буквенных символов не следует добавлять в таблицу CONTENT CHARACTER CHUNK. Как в пятом, так и в шестом битах содержатся значения '0', обозначая, что формат кодирования блока – UTF-8 (см.п. С.20.3.1). В седьмом и восьмом битах содержатся значения '1' и '0' соответственно, обозначая, что длина октетов закодированных символов UTF-8 (блок единиц информации **character**) больше либо равна 3_{10} октетам и меньше либо равна 258_{10} октетам, и что длина, за вычетом нижней границы, кодируется в восьми битах последующего октета как целое число без знака (см.п.С.24.3.2). Октет, стоящий на позиции 230_{16}, со значением 07_{16}, представляет собой это целое число без знака. Длина октетов закодированных символов UTF-8 равна 10_{10} (нижняя граница равна 3_{10}). 10_{10} октетов закодированных символов UTF-8 кодируются, начиная с октета, стоящего на позиции 231_{16} и заканчивая октетом, стоящим на позиции $23a_{16}$.	единицы информации **character** "Marsh Lane"
23b	1111**0000** (f0)	Этот октет является указателем конца для единицы информации **StreetName** типа **element**.	**</cbc:StreetName>**
23c 23d 23e 23f 240 247	00111111 (3f) 10000010 (82) 10000010 (82) 00000111 (07) 01000011 (43) 01100101 (65)	Эти октеты представляют собой кодирование единицы информации **CityName** типа **element**. Свойство **[local name]** "CityName" добавляется в таблицу LOCAL NAME (с индексом, равным 10_{10}). Уточненное имя со значением свойства **[prefix]** "cbc" (индекс 3_{10}), значением свойства **[namespace name]** "……BasicComponents:1:0" (индекс 3_{10}), и значением свойства **[local name]** "CityName" (индекс 11_{10}) добавляется в таблицу ELEMENT NAME (с индексом, равным 10_{10}).	**<cbc:CityName>**
248 249 24a 250	10000010 (82) 00000100 (04) 01001110 (4e) 01100101 (65)	Эти октеты представляют собой кодирование единиц информации **character** данной единицы информации. 7_{10} октетов закодированных символов UTF-8 кодируются, начиная с октета, стоящего на позиции $24a_{16}$ и заканчивая октетом, стоящим на позиции 250_{16}.	единицы информации **character** "Nowhere"
251	1111**0000** (f0)	Этот октет является указателем конца для единицы информации **CityName** типа **element.**	**</cbc:CityName>**
252 253 254 255 256 .... 25f	00111111 (3f) 10000010 (82) 10000010 (82) 00001001 (09) 01000011 (50) 01100101 (65)	Эти октеты представляют собой кодирование единицы информации **PostalZone** типа **element**. Свойство **[local name]** "PostalZone" добавляется в таблицу LOCAL NAME (с индексом, равным 12_{10}). Уточненное имя со значением свойства **[prefix]** "cbc" (индекс 3_{10}), значением свойства **[namespace name]** "……BasicComponents:1:0" (индекс 3_{10}), и значением свойства **[local name]** "PostalZone" (индекс 12_{10}) добавляется в таблицу ELEMENT NAME (с индексом, равным 11_{10}).	**<cbc:PostalZone>**
260 261 262	10000010 (82) 00000101 (05) 01001110 (4e)	Эти октеты представляют собой кодирование единиц информации **character** единицы информации **PostalZone**. 8_{10} октетов закодированных символов UTF-8 кодируются, начиная с октета, стоящего на позиции 262_{16} и заканчивая октетом, стоящим на позиции 269_{16}.	единицы информации **character** "NR18 4XX"

Таблица D.12 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
269	01011000 (58)		
26a	1111<u>0000</u> (f0)	Этот октет является указателем конца для единицы информации **PostalZone** типа **element.**	**</cbc:PostalZone>**
26b 26c 26d 26e 26f 27e	00111111 (3f) 10000010 (82) 10000010 (82) 00001111 (0f) 01000011 (43) 01111001 (79)	Эти октеты представляют собой кодирование единицы информации **CountrySubentity** типа **element**. Свойство **[local name]** "CountrySubentity" добавляется в таблицу LOCAL NAME (с индексом, равным 13_{10}). Уточненное имя со значением свойства **[prefix]** "cbc" (индекс 3_{10}), значением свойства **[namespace name]** "…….BasicComponents:1:0" (индекс 3_{10}), и значением свойства **[local name]** "CountrySubentity" (индекс 13_{10}) добавляется в таблицу ELEMENT NAME (с индексом, равным 12_{10}).	**<cbc:CountrySubentity>**
27f 280 281 287	10000010 (82) 00000100 (04) 01001110 (4e) 01101011 (6b)	Эти октеты представляют собой кодирование единиц информации **character** единицы информации **CountrySubentity**. 7_{10} октетов закодированных символов UTF-8 кодируются, начиная с октета, стоящего на позиции 281_{16} и заканчивая октетом, стоящим на позиции 287_{16}.	единицы информации **character** "Norfolk"
288	11111111 (ff)	Этот октет является указателем конца для единицы информации **CountrySubentity** типа **element** и для единицы информации **Address** типа **element.** В первых четырех битах октета (биты с первого по четвертый), стоящего на позиции 288_{16}, со значением ff_{16}, содержатся значения '1111' (указатель конца) и эти биты являются указателем конца для единицы информации **CountrySubentity** типа **element** (см.п. C.3.8). В последних четырех битах (биты с пятого по восьмой) содержатся значения '1111' и эти биты являются указателем конца для единицы информации **Address** типа **element** (см.п. C.3.8).	**</cbc:CountrySubentity>** **</cac:Address>**

D.5.2.3 Кодирование единицы информации BuyersID типа element из первой единицы информации LineItem типа element

Ниже приводится детальное объяснение кодирования единицы информации **BuyersID** типа **element** из первой единицы информации **LineItem** типа **element** документа Быстрого информационного набора. В частности, объясняется кодирование единицы информации **element** свойство **[local name]** которой было проиндексировано до объяснения данной единицы информации. В Таблице D.13 представлен фрагмент документа Быстрого информационного набора для кодирования единицы информации **BuyersID** типа **element** из первой единицы информации **LineItem** типа **element** из п. D.3.2. В Таблице D.14 приводится подробное описание кодирования. Этот фрагмент в формате XML 1.0 выглядит следующим образом:

```
<cac:LineItem>
    <cac:BuyersID>A</cac:BuyersID>
```

Таблица D.13 – Октеты фрагмента (как шестнадцатеричные символы)

	000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
0003c0	<u>3f</u>8383074c696e654974656d<u>3f</u>8383829041<u>f0</u>

Таблица D.14 – Подробное описание кодирования

	Октет(октеты)	Описание	Информационный набор XML или XML
3c4 3c5 3c6 3c7 3c8 3cf	00111111 (3f) 10000011 (83) 10000011 (83) 00001111 (07) 01001010 (4c) 01101101 (6d)	Эти октеты представляют собой кодирование единицы информации **LineItem** типа **element**. Свойство **[local name]** "LineItem" добавляется в таблицу LOCAL NAME (с индексом, равным 21_{10}). Уточненное имя со значением свойства **[prefix]** "cac" (индекс 4_{10}), значением свойства **[namespace name]** "……ateComponents:1:0" (индекс 4_{10}), и значением свойства **[local name]** "LineItem" (индекс 21_{10}) добавляется в таблицу ELEMENT NAME (с индексом, равным 20_{10}).	**<cac:LineItem>**
3d0 3d1 3d2 3d3	00111111 (3f) 10000011 (83) 10000011 (83) 10000010 (82)	Эти октеты представляют собой кодирование единицы информации **BuyersID** типа **element**. Свойства **[prefix]**, **[namespace name]** и **[local name]** уже были проиндексированы, поскольку связанные с ними сроки все уже встречались ранее до этой единицы информации. Свойство **[local name]** было проиндексировано в ходе обработки первого дочернего элемента единицы информации **Order** типа **element**, а именно единицы информации **BuyersID** типа **element** со значением свойства **[namespace name]** "….Order:1:0". Уточненное имя со значением свойства **[prefix]** "cac" (индекс 4_{10}), значением свойства **[namespace name]** "……ateComponents:1:0" (индекс 4_{10}), и значением свойства **[local name]** "BuyersID" (индекс 3_{10}) добавляется в таблицу ELEMENT NAME (с индексом, равным 21_{10}).	**<cac:BuyersID>**
3d4 3d5	10010000 (90) 01000001 (41)	Эти октеты представляют собой кодирование единиц информации **character** единицы информации **BuyersID** типа **element**. В первых двух битах октета (биты с первого по второй), стоящего на позиции $3d4_{16}$, со значением 90_{16}, содержатся значения '10' (идентификация) обозначая, что у единицы информации **element** есть дочерний элемент (дочерний элемент единицы информации **BuyersID** типа **element**), и этот дочерний элемент является блоком единиц информации **character** (см.п. C.3.7.5). В третьем бите содержится значение '0', обозначая, что присутствует строка буквенных символов (см.п. C.15.3). В четвертом бите содержится значение '1', обозначая, что эту строку буквенных символов следует добавить в таблицу CONTENT CHARACTER CHUNK (в данном примере строки длиной менее 6_{10} символов добавляются в таблицу CONTENT CHARACTER CHUNK или в таблицу ATTRIBUTE VALUE). Как в пятом, так и в шестом битах содержатся значения '0', обозначая, что формат кодирования данного блока – UTF-8 (см.п. C.20.3.1). В седьмом бите содержится значение '0', обозначая, что длина октетов закодированных символов UTF-8 (блок единиц информации **character**) больше либо равна 1_{10} октету и меньше либо равна 2_{10} октетам, и что длина, за вычетом нижней границы, кодируется в восьмом бите как целое число без знака (см. C.24.3.1). Целое число без знака равно 0_{10} и длина равна 1_{10}. Октет закодированных символов UTF-8 кодируется в октете, стоящем на позиции 41_{16}.	единица информации **character** "A"
3d6	11110000 (f0)	Этот октет является указателем конца для единицы информации **BuyersID** типа **element**.	**</cac:BuyersID>**

Приложение E

Назначение значение идентификаторов объектов

(Данное Приложение не является неотъемлемой частью настоящей Рекомендации | Международного стандарта)

В рамках настоящей Рекомендации | Международного стандарта назначены 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" (Модуль ASN.1 Быстрого информационного набора)

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

"Fast Infoset Encoding Definition Module" (Модуль определения кодирования Быстрого информационного набора)

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

"Fast Infoset Encoding Link Module" (Модуль связи кодирования Быстрого информационного набора)

{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 п. A.1*

"A fast infoset document containing an XML declaration" (Документ Быстрого информационного набора, содержащий декларацию XML)

{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 п. A.1*

"A fast infoset document without an XML declaration" (Документ Быстрого информационного набора, не содержащий декларации XML)

СПРАВОЧНАЯ ЛИТЕРАТУРА

[1] OASIS *Universal Business Language (UBL) 1.0.*

[2] IETF RFC 1952 (1996), *GZIP file format specification version 4.3.*

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ-Т

Серия A	Организация работы МСЭ-Т
Серия D	Общие принципы тарификации
Серия E	Общая эксплуатация сети, телефонная служба, функционирование служб и человеческие факторы
Серия F	Нетелефонные службы электросвязи
Серия G	Системы и среда передачи, цифровые системы и сети
Серия H	Аудиовизуальные и мультимедийные системы
Серия I	Цифровая сеть с интеграцией служб
Серия J	Кабельные сети и передача сигналов телевизионных и звуковых программ и других мультимедийных сигналов
Серия K	Защита от помех
Серия L	Конструкция, прокладка и защита кабелей и других элементов линейно-кабельных сооружений
Серия M	Управление электросвязью, включая СУЭ и техническое обслуживание сетей
Серия N	Техническое обслуживание: международные каналы передачи звуковых и телевизионных программ
Серия O	Требования к измерительной аппаратуре
Серия P	Качество телефонной передачи, телефонные установки, сети местных линий
Серия Q	Коммутация и сигнализация
Серия R	Телеграфная передача
Серия S	Оконечное оборудование для телеграфных служб
Серия T	Оконечное оборудование для телематических служб
Серия U	Телеграфная коммутация
Серия V	Передача данных по телефонной сети
Серия X	**Сети передачи данных, взаимосвязь открытых систем и безопасность**
Серия Y	Глобальная информационная инфраструктура, аспекты межсетевого протокола и сети последующих поколений
Серия Z	Языки и общие аспекты программного обеспечения для систем электросвязи

Отпечатано в Швейцарии
Женева, 2007