Network Working Group
Request for Comments: 3629
STD: 63
Obsoletes: 2279
Category: Standards Track

F. Yergeau
Alis Technologies

November 2003

UTF-8, a transformation format of ISO 10646

UTF-8 - преобразование формата ISO 10646

Статус документа

В этом документе содержится спецификация протокола, предложенного сообществу Internet. Документ служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола можно узнать из «Internet Official Protocol Standards» (STD 1). Документ может распространяться без ограничений.

Аннотация

Стандарт ISO/IEC 10646-1 определяет большой набор, называемый универсальным набором символов (Universal Character Set или UCS), который охватывает большинство мировых систем письма. Однако первоначально предложенные кодировки UCS оказались не совместимыми со многими имеющимися приложениями и протоколами, что привело к разработке кодировки UTF-8, описанной в этом документе. Особенностью UTF-8 является сохранение всего диапазоны US-ASCII, что обеспечивает совместимость с файловыми системами, синтаксическими анализаторами и другими программами, основанными на US-ASCII и воспринимающими другие значения. Этот документ заменяет собой RFC 2279.

Оглавление

1. Введение

Стандарт ISO/IEC 10646-1 [ISO.10646] определяет большой набор, называемый универсальным набором символов (UCS), который охватывает большинство мировых систем письма. Этот же набор символов определён стандартом Unicode [UNICODE], задающим дополнительные свойства символов и другие детали применения, важные для разработчиков. До настоящего времени изменения в Unicode, а также поправки и дополнения к ISO/IEC 10646 отслеживают друг друга, так что наборы символов и их коды оставались синхронизированными. Соответствующие комитеты по стандартизации обязались поддерживать эту очень полезную синхронность.

ISO/IEC 10646 и Unicode задают несколько форм кодирования их общего набора - UTF-8, UCS-2, UTF-16, UCS-4 UTF-32. В формате кодирования каждый символ представляется одной или несколькими единицами кодирования. Все стандартные формы кодирования UCS, за исключением UTF-8, используют единицу кодирования больше 1 октета, что затрудняет их применение во многих современных приложениях и протоколах, предполагающих использование 8- или 7-битовых символов.

В UTF-8 - предмете этого документа - используется 1-октетный блок кодирования. Используются все биты октета, но сохраняется весь диапазон US-ASCII [US-ASCII] - символы US-ASCII кодируются в 1 октет с нормальным значением US-ASCII и любой октет с таким значением может обозначать только символ US-ASCII и ничего другого.

UTF-8 представляет символы UCS разным числом октетов, где количество октетов и значение каждого из них зависит от целочисленного кода символа в ISO/IEC 10646 (номер символа, он же кодовая позиция, кодовая точка или скалярное значение Unicode). Характеристики этой формы кодирования приведены ниже в шестнадцатеричной форме.

- Номера символов от U+0000 до U+007F (US-ASCII) соответствуют октетам от 00 до 7F (7-битовые символы US-ASCII). Прямым следствием этого является то, что строка ASCII является действительной строкой UTF-8.
- Значения октетов US-ASCII не появляются иначе в потоке символов с кодировкой UTF-8. Это обеспечивает совместимость с файловыми системами и другими программами (например, с функцией printf() в библиотеках C), которые анализируют значения US-ASCII, и прозрачность для других значений.
- Преобразование между UTF-8 и другими формами кодирования выполняется легко.
- Первый октет многооктетной последовательности указывает число октетов в последовательности.
- Октеты со значениями C0, C1, F5 - FF никогда не используются.
- Границы символов легко найти в любом месте потока октетов.
- Лексический порядок сортировки строк UTF-8 по значениям байтов совпадает с порядком при сортировке по номерам символов. Это представляет ограниченный интерес, поскольку сортировка по номерам символов почти никогда не применяется.
- Алгоритм Бойера-Мура (Boyer-Moore) может применяться для быстрого поиска с данными UTF-8.
- Строки UTF-8 могут достаточно надёжно распознаваться простым алгоритмом. т. е. вероятность того, что строка в другой кодировке будет воспринята как действительная строка UTF-8, мала и уменьшается с ростом длины строки.

Кодировка UTF-8 была разработана в сентябре 1992 г. Ken Thompson на основе критериев, заданных Rob Pike, с целью задания формата преобразования UCS, который можно было бы использовать в операционной системе Plan9 без нарушения её работы. Проект Thompson прошёл стандартизацию в X/Open Joint Internationalization Group (XOJIG, см. [FSS_UTF]), получив имена FSS-UTF (вариант FSS/UTF), UTF-2 и, наконец, UTF-8.

2. Соглашения по обозначениям

Ключевые слова **должно** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с [RFC2119].

Символы UCS обозначаются нотацией U+HHHH, где HHHH - строка из 4 - 6 шестнадцатеричных цифр, представляющая номер символа в ISO/IEC 10646.

3. Определение UTF-8

Определение UTF-8 дано в стандарте Unicode [UNICODE]. Описания и формулы даны также в приложении Annex D к ISO/IEC 10646-1 [ISO.10646]

В UTF-8 символы из диапазона U+0000 - U+10FFFF (доступный диапазон UTF-16) кодируются последовательностями размером от 1 до 4 октетов. В единственном октете однооктетной «последовательности» первый бит сброшен (0), а оставшиеся 7 указывают номер символа. В последовательности из n октетов (n>1) в начальном октете n старших битов имеют значение 1, затем один бит сброшен (0), а остальные содержат биты номера кодируемого символа. В последующих октетах старший бит установлен (1), следующий сброшен (0), а оставшиеся 6 битов кодируют символ. В приведённой ниже таблице показаны форматы этих октетов (x указывает биты, доступные для кодирования символов).

Диапазон шестнадцатеричных значений	***Двоичная последовательность октетов UTF-8***
0000 0000-0000 007F	0xxxxxxx
0000 0080-0000 07FF	110xxxxx 10xxxxxx
0000 0800-0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000-0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Кодирование символа UTF-8 описано ниже.

1. Определяется число требуемых октетов по номеру символа и первому столбцу приведённой ваше таблицы. Важно отметить, что строки таблицы являются взаимоисключающими, т. е. для любого данного символа пригодна лишь 1 строка.
2. Подготавливаются старшие биты октетов в соответствии со вторым столбцом приведённой выше таблицы.
3. Заполняются биты x битами номера символа, представленного в двоичной форме. Сначала размещается младший бит номера в младшем бите последнего октета, затем следующий по старшинству бит этого октета и т. д. Когда биты x последнего октета заполнены, выполняется заполнение предыдущего октета и т. д. до полного заполнения битов x.

Определение UTF-8 запрещает кодирование символов с номерами U+D800 - U+DFFF, которые зарезервированы для использования в UTF-16 (как суррогатные пары) и не представляют символы напрямую. При перекодировании в UTF-8 из UTF-16 нужно сначала декодировать данные UTF-16 для получения номеров символов, которые затем кодируются в UTF-8, как описано выше. Это отличается от кодировки CESU-8 [CESU-8], которая похожа на UTF-8, но не предназначена для Internet. CESU-8 работает аналогично UTF-8, но вместо номеров символов (кодовых точек) кодирует значения кодов UTF-16 (16-битовые величины). Это ведёт к разным результатам для символов с номерами больше 0xFFFF (кодировка CESU-8 для этих символов **недействительна** в UTF-8).

Ниже описано декодирование символов UTF-8.

1. Инициируется двоичное число сбросом всех битов в 0. Может потребоваться до 21 бита.
2. Определяется, какие биты кодируют номер символа, по числу октетов в последовательности и второй колонке приведённой выше таблицы (биты x).

3. Биты x из последовательности копируются в двоичное число, начиная с младшего бита последнего октета последовательности. Когда биты x закончатся, двоичное число будет представлять номер символа.

Реализации этого алгоритма декодирования **должны** обеспечивать защиту от декодирования недопустимых последовательностей. Например, наивная реализация может декодировать чрезмерно длинную последовательность UTF-8 C0 80 как символ U+0000 или суррогатную пару ED A1 8C ED BE B4 - в U+233B4. Декодирование непригодных последовательностей может иметь последствия для безопасности или вызывать другие проблемы (см. раздел 10).

4. Синтаксис последовательностей байтов UTF-8

Для удобства разработчиков, применяющих ABNF, ниже приведено определение UTF-8 в синтаксисе ABNF. Строка UTF-8 - это последовательность октетов, представляющих последовательность символов UCS. Последовательность октетов действительна в UTF-8 лишь в том случае, когда она соответствует представленному ниже синтаксису, выведенному из правил кодирования UTF-8 и выраженному в форме ABNF [RFC2234].

```
UTF8-octets = *( UTF8-char )
UTF8-char   = UTF8-1 / UTF8-2 / UTF8-3 / UTF8-4
UTF8-1      = %x00-7F
UTF8-2      = %xC2-DF UTF8-tail
UTF8-3      = %xE0 %xA0-BF UTF8-tail / %xE1-EC 2( UTF8-tail ) /
              %xED %x80-9F UTF8-tail / %xEE-EF 2( UTF8-tail )
UTF8-4      = %xF0 %x90-BF 2( UTF8-tail ) / %xF1-F3 3( UTF8-tail ) /
              %xF4 %x80-8F 2( UTF8-tail )
UTF8-tail   = %x80-BF
```

Примечание. Официальное определение UTF-8 дано в [UNICODE]. Считается, что приведённая грамматика описывает то же, что и Unicode, но она не претендует на официальность. Разработчикам рекомендуется полагаться на официальный источник, а не на приведённую здесь форму ABNF.

5. Версии стандартов

Стандарт ISO/IEC 10646 время от времени обновляется путём публикации поправок и дополнительных частей, для стандарта Unicode с течением времени тоже публикуются новые версии. Каждая новая версия заменяет собой предыдущую, но реализации и, что важнее всего, данные не обновляются мгновенно.

В общем случае изменения сводятся к добавлению новых символов, что не создаёт проблем со старыми данными. В 1996 г. поправки Amendment 5 к редакции ISO/IEC 10646 1993 г. и Unicode 2.0 перенесли и расширили блок корейского хангыля (Hangul), сделав все прежние данные с символами Hangul непригодными в новой версии. Unicode 2.0 отличается в том же от Unicode 1.1. Оправданием для такого несовместимого изменения послужило отсутствие крупных реализаций и значительных объёмов данных, содержащих Hangul. Этот инцидент был назван корейским беспорядком (Korean mess) и соответствующие комитеты пообещали, никогда впредь не вносить несовместимые изменения (см. правила консорциума Unicode [1]).

Новые версии и в особенности несовместимые изменения влияют на метки MIME, рассматриваемые в разделе 8.

6. Знак порядка байтов (BOM)

Символ UCS U+FEFF ZERO WIDTH NO-BREAK SPACE[1], неформально называемый также BYTE ORDER MARK[2] (BOM) можно действительно применять в качестве пробела нулевой ширины без разрыва, но имя BOM намекает на второе возможное использование символа - добавление U+FEFF в начало потока символов UCS в качестве «подписи». Получатель такого сериализованного потока может использовать его начальный символ как подсказку о содержании в потоке символов UCS, а также для распознания задействованной кодировки символов UCS и при кодировке с многооктетными блоками кодирования в качестве способа распознания порядка сериализации октетов. В UTF-8 применяется однооктетный блок кодирования, поэтому последняя функция бесполезна здесь и BOM всегда будет появляться в виде последовательности октетов EF BB BF.

Важно понимать, что символ U+FEFF в любой позиции, кроме начала потока, **должен** интерпретироваться с семантикой пробела нулевой ширины без разрыва и его **недопустимо** считать подписью. При интерпретации как подписи стандарт Unicode предполагает, сто начальный символ U+FEFF может быть исключён (вырезан) перед обработкой текста. Такое исключение необходимо в некоторых случаях (например, при конкатенации двух строк, поскольку иначе полученная строка будет содержать непредусмотренный неразрывный пробел в точке соединения) но может повлиять на внешний процесс на другом уровне (например, на цифровую подпись или подсчёт символов), который учитывает все символы потока. Поэтому **рекомендуется** избегать вырезания без уважительной причины начального символа U+FEFF, интерпретируемого как подпись, игнорировать его без вырезания, когда это возможно (например, при отображении) и вырезать символ только при реальной необходимости.

U+FEFF в первой позиции потока **может** интерпретироваться не только как подпись, но и как неразрывный пробел нулевой ширины. В попытке сократить неопределённость в Unicode 3.2 добавлен новый символ U+2060 WORD JOINER[3] с такой же семантикой как у U+FEFF, но без функции подписи, и настоятельно рекомендуется применять его для выражения семантики слияния слов. В конечном итоге, соблюдение этой рекомендации практически гарантирует, что любой начальный символ U+FEFF является подписью, а не ZERO WIDTH NO-BREAK SPACE.

Тем не менее, неопределённость сохраняется и может влиять на протоколы Internet. Спецификации протоколов могут ограничивать применение U+FEFF в качестве подписи, чтобы сократить или устранить возможные вредные последствия этой неопределённости. В целях соблюдения баланса преимуществ (снижение неопределённости) и недостатков (потеря функции подписи) таких ограничений полезно различать указанные ниже случаи.

- Протоколам **следует** запрещать использование U+FEFF в качестве подписи для текстовых элементов протокола, которые, согласно протоколу, всегда должны быть в UTF-8, поскольку функция подписи в таких случаях совершенно бесполезна.

[1]Пробел нулевой ширины без разрыва (строки) или просто неразрывный пробел.
[2]Знак порядка байтов.
[3]Объединитель слов.

- Протоколам **следует** запрещать использование U+FEFF в качестве подписи для текстовых элементов протокола, которым протокол обеспечивает механизмы идентификации кодировки, когда ожидается, что реализации протокола будут в состоянии всегда использовать эти механизмы должным образом. Это будет выполняться, если элементы протокола находятся под строгим контролем реализации с момента их создания до момент передачи (с правильной маркировкой).

- Протоколам **не следует** запрещать использование U+FEFF в качестве подписи для текстовых элементов протокола, которым протокол не обеспечивает механизмы идентификации кодировки, когда запрет не может быть применён или предполагается, что реализации протоколов не всегда могут должным образом использовать эти механизмы. Два последних случая, скорей всего, будут возникать для больших элементов протокола, таких как сущности MIME, особенно при получении реализацией протокола таких сущностей от файловых систем, протоколов без механизмов идентификации кодировки содержимого (например, FTP) или иных протоколов, не гарантирующих корректного указания кодировки символов (например, HTTP).

Когда протокол запрещает использовать U+FEFF в качестве подписи для некоторых элементов протокола, любой начальный символ U+FEFF **должен** считаться неразрывным пробелом нулевой ширины. Если протокол **не** запрещает использование U+FEFF в качестве подписи для некоторых элементов протокола, реализациям **следует** быть готовыми к обработке подписи в этом элементе и реагировать должным образом - использовать подпись для идентификации кодировки символов, если это требуется, и удалять или игнорировать подпись, когда это приемлемо.

7. Примеры

Последовательность символов U+0041 U+2262 U+0391 U+002E (A<NOT IDENTICAL TO><ALPHA>.[1]) в кодировке UTF-8 имеет вид

```
--+--------+-----+--
41 E2 89 A2 CE 91 2E
--+--------+-----+--
```

Последовательность символов U+D55C U+AD6D U+C5B4 (Korean "hangugeo" - корейский язык) в кодировке UTF-8 имеет вид

```
--------+--------+--------
ED 95 9C EA B5 AD EC 96 B4
--------+--------+--------
```

Последовательность символов U+65E5 U+672C U+8A9E (Japanese "nihongo" - японский язык) в кодировке UTF-8 имеет вид

```
--------+--------+--------
E6 97 A5 E6 9C AC E8 AA 9E
--------+--------+--------
```

Символ U+233B4 (пень дерева по китайски) с UTF-8 BOM перед ним в кодировке UTF-8 имеет вид

```
--------+-----------
EF BB BF F0 A3 8E B4
--------+-----------
```

8. Регистрация MIME

Этот документ служит основой для регистрации параметра MIME charset для кодировки UTF-8 в соответствии с [RFC2978]. Параметр имеет значение UTF-8. Эта строка помечает типы носителей, содержащие текст из символов набора ISO/IEC 10646 с учётом всех поправок, по меньшей мере, до поправки 5 1993 г. (корейский блок), представленных последовательностями октетов с использованием описанной выше схемы кодирования. UTF-8 подходит для использования в типах содержимого MIME с типом верхнего уровня text.

Примечательно, что метка UTF-8 не включает номера вервии, указывая ISO/IEC 10646 в целом. Это обусловлено тем, что метка набора символов MIME предназначена для представления лишь сведений, требуемых для преобразования последовательности полученных из линии байтов в последовательность символов и не более того (см. параграф 2.2 в [RFC2045]). Пока стандарт для наборов символов не меняется несовместимым образом, номер версии не нужен, поскольку сведения из такого тега о возможности получения недавно добавленных символов, которые не известны, ничего не дают. Сам тег ничего не сообщает о новых символах, которые в любом случае будут получены.

Таким образом, пока стандарты развиваются согласованно, польза от добавления номера версии остаётся лишь видимостью. Но у зависящих от версии меток имеется и недостаток - когда старое приложение получает данные с новой (неизвестной) меткой, оно может не распознать её и оказаться неспособным обработать данные, тогда как базовая (известная) метка приведёт к корректной в основном обработке данных, которые могут не включать новых символов.

Сейчас «корейский беспорядок» (ISO/IEC 10646, поправка 5) является несовместимым изменением, в принципе противоречащим целесообразности независимой от версии метки набора символов MIME, как описано выше. Однако проблема совместимости возможна лишь для данных с символами Korean Hangul, закодированными в соответствии с Unicode 1.1 (или, эквивалентно, до поправки 5 к ISO/IEC 10646), а таких данных, пожалуй, не существует и именно по этой причине несовместимое изменение было сочтено приемлемым.

Таким образом, на практике оправдана метка без указания версии при условии, что она относится ко всем версиям после Amendment 5 и несовместимых изменений не происходит. Если в более поздней версии ISO/IEC 10646 будут несовместимые изменения, метка набора символов MIME останется согласованной с предыдущей версией, пока IETF не примет соответствующее решение.

9. Взаимодействие с IANA

В записи для UTF-8 в реестре IANA для наборов символов указана ссылка на этот документ.

[1] A не идентично ALPHA.

10. Вопросы безопасности

Разработчикам UTF-8 нужно учитывать аспекты безопасности при обработке недопустимых последовательностей UTF-8. Вполне возможно, что в некоторых условиях злоумышленник сможет воспользоваться неосторожным синтаксическим анализатором UTF-8, передав тому последовательность октетов, не разрешённую синтаксисом UTF-8. Особо тонкий вариант такой атаки можно организовать на анализатор, который выполняет важные в плане безопасности проверки входных данных в кодировке UTF-8, считая символами некоторые недопустимые последовательности октетов. Например, анализатор может запрещать символ NUL, представленный однооктетной последовательностью 00, но ошибочно разрешать некорректную двухоктетную последовательность C0 80, считая её символом NUL. Другим примером может служить анализатор, запрещающий последовательность 2F 2E 2E 2F (/../), но разрешающий 2F C0 AE 2E 2F. Этот эксплойт был использован в широко распространённом вирусе, атаковавшем Web-серверы в 2001 г, т. е. угроза вполне реальны.

Ещё одна проблема возникает при кодировании UTF-8 - описание ISO/IEC 10646 для UTF-8 разрешает кодировать номера символов вплоть до U+7FFFFFFF, что даёт последовательности размером до 6 байтов. Поэтому возникает риск переполнения буфера, если диапазон номеров символов явно не ограничен значением U+10FFFF или размер буфера не учитывает возможность использования последовательностей из 5 или 6 байтов.

На безопасность может влиять возможность в некоторых кодировках (включая UTF-8) представить одно и то же (насколько может судить пользователь) разными последовательностями символов. Например, букву e с сильным ударением (acute accent) можно представить составным символом U+00E9 E ACUTE или канонически эквивалентной последовательностью U+0065 U+0301 (E + COMBINING ACUTE). Хотя UTF-8 предоставляет одну последовательность байтов для каждой последовательности символов, наличие нескольких последовательностей символов для «одного и того же» может иметь последствия для безопасности при сопоставлении строк, индексировании, поиске, сортировке, сопоставлении регулярных выражений и выборе. Примером может служить сопоставления строк идентификаторов в свидетельствах (credential) и записях списка контроля доступа. Эту проблему можно решить с помощью форм нормализации Unicode, см. [UAX15].

11. Благодарности

В подготовке и обсуждении этого документа принимали участие James E. Agenbroad, Harald Alvestrand, Andries Brouwer, Mark Davis, Martin J. Duerst, Patrick Faltstrom, Ned Freed, David Goldsmith, Tony Hansen, Edwin F. Hart, Paul Hoffman, David Hopwood, Simon Josefsson, Kent Karlsson, Dan Kohn, Markus Kuhn, Michael Kung, Alain LaBonte, Ira McDonald, Alexey Melnikov, MURATA Makoto, John Gardiner Myers, Chris Newman, Dan Oscarsson, Roozbeh Pournader, Murray Sargent, Markus Scherer, Keld Simonsen, Arnold Winkler, Kenneth Whistler и Misha Wolf.

12. Отличия от RFC 2279

- Набор символов ограничен диапазоном 0000-10FFFF (диапазон, доступный в UTF-16).
- Стандарт Unicode сделан источником нормативного определения UTF-8, а ISO/IEC 10646 оставлен как справочник по символам.
- Уточнена терминология. UTF-8 описывается в терминах формы кодирования номера символа, а UCS-2 и UCS-4 практически исчезли.
- Примечание о декодировании непригодных последовательностей стало нормативным требованием **недопустимо** (MUST NOT).
- Добавлен новый раздел о UTF-8 BOM с рекомендациями для протоколов.
- Удалена предложенная регистрация UNICODE-1-1-UTF-8 MIME.
- Добавлен синтаксис ABNF для допустимых последовательностей октетов UTF-8.
- Расширен раздел вопросов безопасности, в частности, рассмотрено влияние нормализации Unicode.

13. Нормативные документы

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[ISO.10646] International Organization for Standardization, "Information Technology - Universal Multiple-octet coded Character Set (UCS)", ISO/IEC Standard 10646, comprised of ISO/IEC 10646-1:2000, "Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane", ISO/IEC 10646-2:2001, "Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 2: Supplementary Planes" and ISO/IEC 10646-1:2000/Amd 1:2002, "Mathematical symbols and other characters".

[UNICODE] The Unicode Consortium, "The Unicode Standard — Version 4.0", defined by The Unicode Standard, Version 4.0 (Boston, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1), April 2003, <http://www.unicode.org/unicode/standard/versions/enumeratedversions.html#Unicode_4_0_0>.

14. Дополнительная литература

[CESU-8] Phipps, T., "Unicode Technical Report #26: Compatibility Encoding Scheme for UTF-16: 8-Bit (CESU-8)", UTR 26, April 2002, <http://www.unicode.org/unicode/reports/tr26/>.

[FSS_UTF] X/Open Company Ltd., "X/Open Preliminary Specification -- File System Safe UCS Transformation Format (FSS-UTF)", May 1993, <http://wwwold.dkuug.dk/jtc1/sc22/wg20/docs/N193-FSS-UTF.pdf>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2978] Freed, N. and J. Postel, "IANA Charset Registration Procedures", BCP 19, RFC 2978, October 2000.

[UAX15] Davis, M. and M. Duerst, "Unicode Standard Annex #15: Unicode Normalization Forms", An integral part of The Unicode Standard, Version 4.0.0, April 2003, <http://www.unicode.org/unicode/reports/tr15>.

[US-ASCII] American National Standards Institute, "Coded Character Set - 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

15. URI

[1] <http://www.unicode.org/unicode/standard/policies.html>

16. Права интеллектуальной собственности

IETF не занимает какой-либо позиции в отношении действительности или объёма каких-либо прав интеллектуальной собственности или иных прав, которые могут быть заявлены как относящиеся к реализации или применению технологии, описанной в этом документе, или степени, в которой любая лицензия, по которой права могут или не могут быть доступны, не заявляется также применение каких-либо усилий для определения таких прав. Сведения о процедурах IETF в отношении прав в документах, связанных со стандартами, можно найти в BCP-11. Копии раскрытия IPR, предоставленные секретариату IETF, и любые гарантии доступности лицензий, а также результаты попыток получить общую лицензию или право на использование таких прав собственности разработчиками или пользователями этой спецификации, можно получить в IETF Secretariat.

IETF предлагает любой заинтересованной стороне обратить внимание на авторские права, патенты или использование патентов, а также иные права собственности, которые могут потребоваться для реализации этого стандарта. Эту информацию следует направлять исполнительному директоу IETF (Executive Director).

17. Адрес автора

Francois Yergeau
Alis Technologies
100, boul. Alexis-Nihon, bureau 600
Montreal, QC H4M 2P2
Canada
Phone: +1 514 747 2547
Fax: +1 514 747 2561
EMail: fyergeau@alis.com

18. Полное заявления авторских прав

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru