Network Working Group
Request for Comments: 4648
Obsoletes: 3548
Category: Standards Track

S. Josefsson
SJD

October 2006

Форматы представления данных Base16, Base32 и Base64

The Base16, Base32, and Base64 Data Encodings

Статус документа

Этот документ содержит спецификацию стандартного протокола, предложенного сообществу Internet, и служит приглашением к дискуссии в целях развития. Текущее состояние стандартизации и статус описанного здесь протокола можно узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться без ограничений.

Аннотация

Этот документ описывает широко используемые схемы представления (кодирования) base 64, base 32 и base 16. Рассмотрено также использование переводов строк в кодированных данных и не включённых в алфавит символов, а также каноническое представление.

Оглавление

1. Введение

Представление base для двоичных данных используется во многих случаях при хранении или переносе данных в средах, которые (возможно, в силу традиций) ограничены использованием кодировки US-ASCII [1]. Такое представление может использоваться также в новых приложениях, не отягощённых традиционными ограничениями, просто потому, что оно позволяет работать с объектами в обычных текстовых редакторах.

В прошлом разные приложения предъявляли различные требования и реализации base-представления несколько отличались. Сегодня спецификации протоколов иногда используют представление base в общем случае и base64 в некоторых частных случаях без точного описания или указания. Обозначение MIME[1] [4] зачастую служит в качестве эквивалента base64 без указания последствий использования переводов строк и не включённых в алфавит символов. Цель этой спецификации заключается в организации общего подхода для алфавита и представления. Это будет существенно снижать влияние неоднозначностей в других документах и повышать уровень совместимости.

2. Используемые соглашения

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с [2].

[1] Multipurpose Internet Mail Extensions - многоцелевые расширения электронной почты Internet.

3. Различия в реализациях

В этом разделе рассматриваются различия в прошлых реализациях представления base и, по возможности, задаётся рекомендуемое для будущих версий поведение.

3.1. Переводы строк в кодированных данных

MIME [4] зачастую указывается в качестве эквивалента base 64, однако спецификация MIME не определяет представление base 64, как таковое, а описывает вариант base 64 Content-Transfer-Encoding[1] для использования в MIME. В связи с этим MIME увеличивает предельный размер строки данных в представлении base с 64 до 76 символов. Расширение MIME наследует представление данных из PEM[2] [3], заявляя о «виртуальной идентичности», однако в PEM используются строки размером 64 символа. Как MIME, так и PEM предназначены для использования в SMTP.

Реализациям **недопустимо** добавлять символы перевода строки в base-представление данных, если спецификация со ссылкой на этот документ явно не говорит о добавлении перевода строки после указанного числа символов в строке.

3.2. Заполнение в кодированных данных

В некоторых случаях заполнение (=) для base-представления данных не требуется и не используется. В общем случае при невозможности оценки размера передаваемых данных нужно использовать заполнение, чтобы декодирование было выполнено корректно.

Реализации **должны** включать подходящие символы заполнения в конце кодированных данных, если спецификация со ссылкой на данный документ явно не утверждает обратное.

Алфавиты base64 и base32 используют заполнение как описано в разделах 4 и 6, но для base16 алфавит не требует заполнения (см. раздел 8).

3.3. Интерпретация не входящих в алфавит символов кодированных данных

Base-кодирование использует специальный, сокращённый алфавит для представления двоичных данных. В закодированных с помощью base данных могут встречаться не включённые в алфавит символы, возникающие в результате повреждения данных или ошибки. Не входящие в алфавит символы могут использоваться в качестве «скрытого канала», через который можно передавать не относящиеся к протоколу данные со злонамеренными целями. Такие символы могут также применяться для того, чтобы воспользоваться ошибками реализации (например, переполнением буферов).

Реализации при преобразовании base-кодированных данных **должны** отвергать кодированные данные при наличии в них не включённых в алфавит символов, если спецификация, основанная на этом документе, явно не указывает иное. Такие спецификации могут указывать, как это сделано в MIME, что символы, не входящие в алфавит представления base следует просто игнорировать при интерпретации данных («будьте снисходительны на приёме»). Отметим, что это может приводить к игнорированию последовательностей символов возврата каретки и перевода строки (CRLF), как не включённых в алфавит. Кроме того, такие спецификации **могут** игнорировать символ заполнения (=), трактуя его как не включенный в алфавит, если он встречается до завершения кодированных данных. При наличии в конце строки более дозволенного числа символов заполнения (например, строка base 64, имеющая в конце последовательность ===) такие избыточные символы также **могут** игнорироваться.

3.4. Выбор алфавита

Разные приложения предъявляют различные требования к символам алфавита. Ниже приведены некоторые требования, определяющие выбор используемого алфавита.

- Обработка человеком. Символы "0" и "O" легко спутать, равно как и "1", "l" и "I". В алфавите base32, показанном ниже, где символы 0 (ноль) и 1 (один) не присутствуют, декодировщик может интерпретировать 0 как O, а 1 как I или L в зависимости от регистра (однако, по умолчанию этого не следует делать, как отмечено в предыдущем параграфе).

- Кодирование в структуры, предписывающие другие требования. Для алфавитов base 16 и base 32 это определяет использование алфавитов со строчными или прописными буквами. Для base 64 могут возникать проблемы с символами, которые не являются буквами или цифрами (в частности, /), в именах файлов и URL.

- Использование в качестве идентификаторов. Некоторые символы (в частности, + и / в алфавите base 64) трактуются средствами поиска и индексирования текстов, как разрывы слов.

Не существует универсального алфавита, который удовлетворял бы всем требованиям. Пример весьма специализированного варианта приведён в IMAP [8]. В этом документе описаны и поименованы некоторые из применяемых в настоящее время алфавитов.

3.5. Каноническое представление

Этап заполнения для представлений base 64 и base 32 при некорректной реализации может приводить к незначительным изменениям кодированных данных. Например, при кодировании единственного октета с помощью 64 будут использованы все 6 битов первого символа и только 2 бита следующего. Эти биты заполнения **должны** быть установлены в 0, соответствующими спецификации кодерами, как указано ниже при описании заполнения. Если этого не делать, представление данных будет отличаться от канонического и множество строк с представлением base может быть декодировано в одну и ту же последовательность двоичных данных. При выполнении этого требования (и других требований этого документа) гарантируется каноническое представление.

В некоторых средах изменения критичны и, следовательно, декодер **может** принять решение об отказе от декодирования, если биты заполнения не установлены в 0. Спецификации, связанные с этим, могут требовать указанного поведения.

[1] Представление base 64 для обмена содержимым.
[2] Privacy Enhanced Mail - приватные почтовые расширения.

4. Представление Base 64

Приведённое ниже описание base 64 создано на основе документов [3], [4], [5] и [6]. Это представление иной раз обозначают также base64.

Кодирование Base 64 разработано для представления произвольных последовательностей октетов в форме, позволяющей использовать строчные и прописные буквы, но не обязательно понятной человеку.

Используется 65-символьное подмножество набора символов US-ASCII, обеспечивающее представление одним печатным символом 6 битов данных (дополнительный 65-й символ используется для обозначения функции специальной обработки).

Процесс кодирования представляет группу из 24 последовательных битов в форме строки из 4 символов. Обработка выполняется слева направо, а 24-битовая исходная группа образуется конкатенацией трёх 8-битовых групп (байтов). Эти 24 бита после этого трактуются, как 4 сцепленных группы по 6 битов, каждая из которых транслируется в один символ алфавита base 64.

Каждая 6-битовая группа используется в качестве индекса массива из 64 печатных символов. Символы алфавита, соответствующие индексу, помещаются в выходную строку.

Таблица 1. Алфавит Base 64.

Значение	***Код***	***Значение***	***Код***	***Значение***	***Код***	***Значение***	***Код***
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	заполнение	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Специальная обработка выполняется в тех случаях, когда последняя группа входных данных содержит меньше 24 битов. Кодируемое значение всегда завершается полным квантом кодирования. Когда на входе доступно менее 24 битов, входная группа дополняется (справа) нулями до формирования целого числа 6-битовых групп. Заполнение в конце данных осуществляется с использованием символа «=». Поскольку входная информация base 64 всегда включает целое число октетов, возможны лишь перечисленные ниже случаи.

(1) Размер финального блока кодирования на входе кратен 24 битам, кодированный результат будет содержать целое число 4-символьных групп без заполнения символами «=».

(2) Размер финального блока кодирования на входе составляет 8 битов, выходной блок будет представлять 2 символа, дополненные последовательностью из двух символов заполнения «==».

(3) Размер финального блока кодирования на входе составляет 16 битов, выходной блок будет представлять 3 символа, дополненные символом заполнения «=».

5. Представление Base 64 с безопасным алфавитом

Кодирование Base 64 с безопасным для представления URL и имён файлов было использовано в [12].

Предлагался вариант алфавита, использующий «~» в качестве 63-го символа. Поскольку символ «~» в некоторых файловых системах имеет специальное значение, вместо этого рекомендуется кодирование, описанное в этом параграфе. Остаётся не зарезервированный символ URI «.», но некоторые файловые системы не допускают наличия в именах файлов множества точек, что делает символ «.» не подходящим.

Для символа заполнения «=» при использовании его в URI обычно применяется процент-кодирование [9], но если размер данных указывается неявно, этого можно избежать, опуская заполнение (см. параграф 3.2).

Описанное здесь кодирование можно обозначить base64url. Это представление не следует считать эквивалентом base64 и не следует обозначать, как base64. Если явно не указано иное, обозначение base64 указывает кодирование base 64, описанное выше.

Процесс кодирования технически идентичен описанному выше за исключением замены символов 62 и 63, как показано в таблице 2.

Таблица 2. Алфавит Base 64, безопасный для имён файлов и URL.

Значение	Код	Значение	Код	Значение	Код	Значение	Код
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	- (минус)
12	M	29	d	46	u	63	_ (подчерк.)
13	N	30	e	47	v		
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y	заполнение	=

6. Представление Base 32

Ниже приведено описание кодирования base 32, основанного на [11] (с корректировками). Это кодирование можно обозначить base32.

Кодирование Base 32 разработано для представления произвольных последовательностей октетов в форме, которая не будет восприимчива к регистру символов и не обязана быть понятной человеку.

Используется 33-символьное подмножество US-ASCII, позволяющее представлять одним символом 5 битов (дополнительный 33-й символ = служит для обозначения специальной обработки).

Процесс кодирования представляет 40-битовую группу входных данных восемью символами на выходе. Обработка выполняется слева направо, а 40-битовая входная группа образуется конкатенацией пяти 8-битовых входных группы. Эти 40 битов далее трактуются, как восемь сцепленных (конкатенация) 5-битовых групп, каждая из которых транслируется в один символ алфавита base 32. При кодировании битового потока с помощью base 32 предполагается передача в потоке сначала старшего бита. Т. е., первый бит потока является старшим в первом байте потока, а восьмой - младшим в этом байте и т. д.

Каждая 5-битовая группа служит индексом массива из 32 печатных символов. Указанный индексом символ помещается в выходную строку. Эти символы, показанные в таблице 3, представляют собой цифры и заглавные буквы US-ASCII.

Таблица 3. Алфавит Base 32.

Значение	Код	Значение	Код	Значение	Код	Значение	Код
0	A	9	J	18	S	27	3
1	B	10	K	19	T	28	4
2	C	11	L	20	U	29	5
3	D	12	M	21	V	30	6
4	E	13	N	22	W	31	7
5	F	14	O	23	X		
6	G	15	P	24	Y	заполнение	=
7	H	16	Q	25	Z		
8	I	17	R	26	2		

Если финальная группа кодируемых данных имеет размер менее 40 битов, выполняется специальная обработка такой группы. Кодируемое значение всегда завершается полным квантом кодирования. Когда на входе доступно менее 40 битов, входная группа дополняется (справа) нулями до формирования целого числа 5-битовых групп. Заполнение в конце данных осуществляется с использованием символа «=». Поскольку входная информация base 32 всегда включает целое число октетов, возможны лишь перечисленные ниже случаи.

(1) Размер финального блока кодирования на входе кратен 40 битам, кодированный результат будет содержать целое число 8-символьных групп без заполнения символами «=».

(2) Размер финального блока кодирования на входе составляет 8 битов, выходной блок будет представлять 2 символа, дополненные последовательностью из 6 символов заполнения «======».

(3) Размер финального блока кодирования на входе составляет 16 битов, выходной блок будет представлять 4 символа, дополненные последовательностью из 4 символов заполнения «====».

(4) Размер финального блока кодирования на входе составляет 24 бита, выходной блок будет представлять 5 символов, дополненных последовательностью из 3 символов заполнения «===».

(5) Размер финального блока кодирования на входе составляет 32 бита, выходной блок будет представлять 7 символов, дополненных символом заполнения «=».

7. Представление Base 32 с расширенным алфавитом

Ниже приведено описание кодирования base 32, основанного на [7]. Это представление можно назвать base32hex. Его не следует считать эквивалентом base32 и не следует обозначать base32. Такое представление используется, например, в NextSECure3 (NSEC3) [10].

Одним из свойств алфавита этого представления, отсутствующим в base64 и base32, является сохранение порядка сортировки кодированной информации при побитовом сравнении.

Кодирование совпадает с предыдущим вариантом и отличается лишь алфавит, показанный в таблице 4.

Таблица 4. Расширенный (Extended Hex) алфавит Base 32.

Значение	Код	Значение	Код	Значение	Код	Значение	Код
0	0	9	9	18	I	27	R
1	1	10	A	19	J	28	S
2	2	11	B	20	K	29	T
3	3	12	C	21	L	30	U
4	4	13	D	22	M	31	V
5	5	14	E	23	N		
6	6	15	F	24	O	заполнение	=
7	7	16	G	25	P		
8	8	17	H	26	Q		

8. Представление Base 16

Приведённое ниже описание является оригинальным, но аналогично предшествующим. Существенно то, что представление Base 16 является стандартным, независимым от регистра символов шестнадцатеричным кодированием и может обозначаться base16 или hex.

В представлении используется 16-символьное подмножество US-ASCII, позволяющее кодировать 4 бита одним печатным символом.

В процессе кодирования входные 8-битовые группы (октеты) представляются последовательностями из 2 символов. Обработка входных данных выполняется слева направо. 8 входных битов рассматриваются как конкатенация двух 4-битовых групп, каждая из которых транслируется в один символ алфавита base 16.

Каждая 4-битовая группа используется в качестве индекса массива из 16 печатных символов. Соответствующий входному индексу символ помещается в строку выхода.

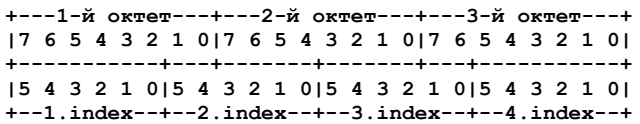
Таблица 5. Алфавит Base 16.

Значение	Код	Значение	Код	Значение	Код	Значение	Код
0	0	4	4	8	8	12	C
1	1	5	5	9	9	13	D
2	2	6	6	10	A	14	E
3	3	7	7	11	B	15	F

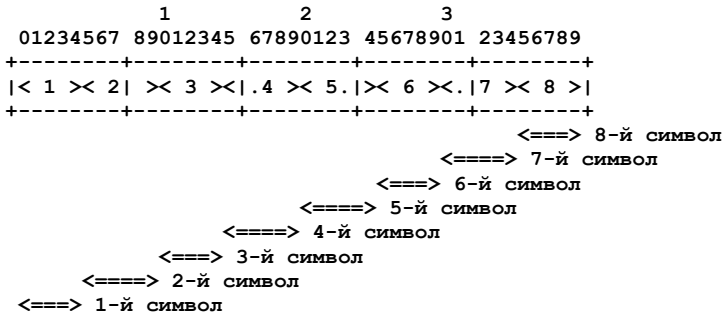
В отличие от base 32 и base 64 символов заполнения не требуется, поскольку всегда доступно полное кодовое слово.

9. Иллюстрации и примеры

Для трансляции двоичных данных в представление base входные данные сохраняются в структуре, а выходные извлекаются. Кодирование base 64 представлено на рисунке, заимствованном из [5].



Кодирование base 32 представлено на рисунке, заимствованном из [7]. Каждый из символов base 32 представляет 5 последовательных битов входных данных. Таким образом, каждая группа из 8 символов представляет последовательность из 5 октетов (40 битов).



Ниже приведён пример данных Base64 из работы [5] с некоторыми корректировками.

```
Входные данные - 0x14fb9c03d97e
Шестнадц.1   4    f   b    9   c     | 0   3    d   9    7   e
8 битов  00010100 11111011 10011100  | 00000011 11011001 01111110
6 битов  000101 001111 101110 011100 | 000000 111101 100101 111110
Десятич. 5      15     46     28       0      61     37     62
Выход    F      P      u      c        A      9      l      +

Входные данные - 0x14fb9c03d9
Шестнадц.1   4    f   b    9   c     | 0   3    d   9
8 битов  00010100 11111011 10011100  | 00000011 11011001
                                         заполнение 00
6 битов  000101 001111 101110 011100 | 000000 111101 100100
Десятич. 5      15     46     28       0      61     36
                                            заполнение =
Выход    F      P      u      c        A      9      k      =

Входные данные - 0x14fb9c03
Шестнадц.1   4    f   b    9   c     | 0   3
8 битов  00010100 11111011 10011100  | 00000011
                                     заполнение 0000
```

```
      6 битов  000101 001111 101110 011100 | 000000 110000
      Десятич. 5      15     46     28       0      48
                                                заполнение =      =
      Выход    F      P      u      c        A      w      =      =
```

10. Тестовые векторы

```
BASE64("") = ""
BASE64("f") = "Zg=="
BASE64("fo") = "Zm8="
BASE64("foo") = "Zm9v"
BASE64("foob") = "Zm9vYg=="
BASE64("fooba") = "Zm9vYmE="
BASE64("foobar") = "Zm9vYmFy"
BASE32("") = ""
BASE32("f") = "MY======"
BASE32("fo") = "MZXQ===="
BASE32("foo") = "MZXW6==="
BASE32("foob") = "MZXW6YQ="
BASE32("fooba") = "MZXW6YTB"
BASE32("foobar") = "MZXW6YTBOI======"
BASE32-HEX("") = ""
BASE32-HEX("f") = "CO======"
BASE32-HEX("fo") = "CPNG===="
BASE32-HEX("foo") = "CPNMU==="
BASE32-HEX("foob") = "CPNMUOG="
BASE32-HEX("fooba") = "CPNMUOJ1"
BASE32-HEX("foobar") = "CPNMUOJ1E8======"
BASE16("") = ""
BASE16("f") = "66"
BASE16("fo") = "666F"
BASE16("foo") = "666F6F"
BASE16("foob") = "666F6F62"
BASE16("fooba") = "666F6F6261"
BASE16("foobar") = "666F6F626172"
```

11. Реализация ISO C99 для представления Base64

Реализация ISO C99 для представления Base64, которая предполагается соответствующей рекомендациям данного документа, доступна по ссылке http://josefsson.org/base-encoding/. Этот код не является нормативным.

Указанный код не включён в данных RFC по процедурным причинам (параграф 5.4 RFC 3978).

12. Вопросы безопасности

При реализации представления base следует соблюдать осторожность, чтобы не создать уязвимостей к атакам с переполнением буферов или иным типам атак. Декодерам не следует прерывать работу по ошибкам на входе, включая, например, символы NUL (ASCII 0).

Если вместо отказа от декодирования происходит игнорирование не входящих в алфавит символов (как рекомендуется), это позволяет организовать скрытый канал, который может служить для утечки информации. Игнорируемые символы могут также применяться с иными неблаговидными целями, например для нарушения операций сравнения строк или активизации ошибок реализации. В приложениях, не следующих установленной практике, следует принимать во внимание результаты игнорирования не включённых в алфавит символов. Аналогично при использовании алфавитов base 16 и base 32, не учитывающих регистр символов, смена регистра может служить для организации скрытого канала утечки или нарушения операций сравнения строк.

При использовании заполнения имеются незначимые биты, которые могут нарушать безопасность за счёт злонамеренного использования с целью организации утечки, нарушения работы операций сравнения или использования ошибок реализации.

Представление Base визуально скрывает информацию (например, пароли), но не обеспечивает защиты конфиденциальности. Это может приводить к нарушениям безопасности - например, пользователь при отправке информации о деталях протокольного обмена (возможно, при устранении ошибок) может нечаянно отправить пароль просто потому, что ему неизвестно о том, что кодирование base не обеспечивает защиты.

Представление Base не увеличивает энтропию текста, но увеличивает его объем и даёт дополнительную информацию для криптоанализа в форме параметров распределения символов.

13. Отличия от RFC 3548

Добавлен расширенный алфавит base32, позволяющий сохранить порядок сортировки для кодированных данных.

Указана ссылка на IMAP для используемого там специального случая кодирования Base64.

Исправлен пример, скопированный из RFC 2440.

Добавлено рассмотрение вопросов безопасности в части обеспечения информации для криптоанализа.

Добавлены тестовые векторы.

Устранены опечатки.

14. Благодарности

Свои комментарии и предложения предоставил ряд людей, включая John E. Hadstate, Tony Hansen, Gordon Mohr, John Myers, Chris Newman и Andrew Sieber. Используемый в документе текст основан на более ранних RFC, описывающих

конкретные варианты разных base-представлений. Автор благодарен RSA Laboratories за поддержку работы, которая привела к созданию этого документа.

Эта переработанная версия частично основана на комментариях и предложениях от Roy Arends, Eric Blake, Brian E Carpenter, Elwyn Davies, Bill Fenner, Sam Hartman, Ted Hardie, Per Hygum, Jelte Jansen, Clement Kent, Tero Kivinen, Paul Kwiatkowski и Ben Laurie.

15. Условия копирования

16. Литература

16.1. Нормативные документы

[1] Cerf, V., "ASCII format for network interchange", RFC 20, October 1969.

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

16.2. Дополнительная литература

[3] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", RFC 1421, February 1993.

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

[5] Callas, J., Donnerhacke, L., Finney, H., and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

[6] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[7] Klyne, G. and L. Masinter, "Identifying Composite Media Features", RFC 2938, September 2000.

[8] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

[9] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[10] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNSSEC Hash Authenticated Denial of Existence", Work in Progress, June 2006.

[11] Myers, J., "SASL GSSAPI mechanisms", Work in Progress, May 2000.

[12] Wilcox-O'Hearn, B., "Post to P2P-hackers mailing list", http://zgp.org/pipermail/p2p-hackers/2001-September/000315.html, September 2001.

Адрес автора

Simon Josefsson

SJD

EMail: simon@josefsson.org

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

IETF не принимает какой-либо позиции в отношении действительности или объема каких-либо прав интеллектуальной собственности (Intellectual Property Rights или IPR) или иных прав, которые, как может быть заявлено, относятся к реализации или использованию описанной в этом документе технологии, 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 RFC можно найти в BCP 78 и BCP 79.

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 из сетевого репозитория IETF IPR по ссылке http://www.ietf.org/ipr.

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. Информацию следует направлять в IETF по адресу ietf-ipr@ietf.org.

Подтверждение

Финансирование функций RFC обеспечено IETF Administrative Support Activity (IASA).