Network Working Group
Internet-Draft
Expires: May 14, 2002

S. Josefsson (editor)
November 13, 2001

Base Encodings
draft-josefsson-base-encoding

Status of this Memo

Copyright Notice

Abstract

This draft contain descriptions of the commonly used base 64, base 32, and base 16 encoding schemes. It also discusses the use of line-feeds in encoded data, use of padding in encoded data, use of non-alphabet characters in encoded data, and use of different encoding alphabets.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [3].

Table of Contents

1. Implementation variances

Base encodings have historically been implemented with some minor variances. This section describe these differences, and mandate a default behaviour, to reduce the possibility for ambiguity in other documents using base encodings. Optimizations, such as those used in PDF's Base 85 encoding, are not discussed.

1.1 Line feeds in encoded data

RFC 2045 [2] is often used as a reference for base 64 encoding. However, RFC 2045 does not define "base 64" per se, but rather a "base 64 Content-Transfer-Encoding" for use within MIME. As such, RFC 2045 enforces a limit on line length of base 64 encode data to 76 characters.

Implementation of specifications using this document as reference for base encodings MUST NOT add line feeds to the encoded data, unless explicitely stated and handled otherwise in said specifications.

1.2 Padding of encoded data

In some circumstances, the use of padding ("=") in base encoded data is not required nor used.

Implementation of specifications using this document as reference for base encodings MUST do proper padding to the encoded data, unless explicitely stated and handled otherwise in said specifications.

1.3 Interpretation of non-alphabet characters in encoded data

Base encodings use a specific, reduced, alphabet to encode binary data. Non base alphabet characters may exist within base encoded data, caused by data corruption or by design.

Implementations of specifications using this document as reference for base encodings MUST ignore characters outside the base encoding alphabet when interpreting base encoded data (``be liberal in what you accept''), unless explicitely stated and handled otherwise in said specifications.

Note that this means that e.g., CRLF-padding after 76 characters constitue "non alphabet characters", and should simply be ignored. Also, the pad character, "=", should not be regarded as part of the base alphabet until the end of the string. If more than the allowed number of pad characters are found at the end of the string, e.g., a base 64 string terminated with "===" the excess pad characters should preferably be ignored in a robust implementation.

1.4 Chosing the alphabet

Different applications have different requirements on the characters in the alphabet. Here are a few requirements that determine which alphabet should be used:

- Handled by humans. Characters "0", "O" are easily interchanged, as well "1", "l" and "I".
- Encoded into structures that place other requirements. This determines the use of upper- or lowercase alphabets (for case-insensitive alphabets such as base 32). For base 64, the non-alphanumeric characters (especially "/") may be problematic in filenames and URLs.
- Used as identifiers. Certain characters, notably "+" and "/" in the base 64 alphabet, are treated as word-breaks by legacy text search/index tools.

There is no universally accepted alphabet that fulfill all the requirements. In this document, we document and name some currently used alphabet variances.

2. Base 64 Encoding

The following description of base 64 is due to [1], [2], [4] and [5]. The URL and filename safe base 64 alphabet is due to [8]. (An alternative alphabet has been suggested as a URL safe alphabet, which used "~" as the 63rd character. However, since this character has special meaning in some file system environments, the "URL and Filename safe" alphabet below is recommended instead.)

The Base 64 encoding is designed to represent arbitrary sequences of octets in a form that requires case sensitivity but need not be humanly readable.

A 65-character subset of US-ASCII is used, enabling 6 bits to be represented per printable character. (The extra 65th character, "=", is used to signify a special processing function.)

The encoding process represents 24-bit groups of input bits as output strings of 4 encoded characters. Proceeding from left to right, a 24-bit input group is formed by concatenating 3 8-bit input groups. These 24 bits are then treated as 4 concatenated 6-bit groups, each

of which is translated into a single digit in the base 64 alphabet.

Each 6-bit group is used as an index into an array of 64 printable characters. The character referenced by the index is placed in the output string.

Table 1: The "Canonical" Base 64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
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	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Table 2: The "URL and Filename safe" Base 64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
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 - (minus)
12 M	29 d	46 u	63 _ (understrike)
13 N	30 e	47 v	
14 O	31 f	48 w	(pad) =
15 P	32 g	49 x	
16 Q	33 h	50 y	

Special processing is performed if fewer than 24 bits are available at the end of the data being encoded. A full encoding quantum is always completed at the end of a quantity. When fewer than 24 input bits are available in an input group, zero bits are added (on the right) to form an integral number of 6-bit groups. Padding at the end of the data is performed using the '=' character. Since all base 64 input is an integral number of octets, only the following cases can arise:

(1) the final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output will be an integral multiple of 4 characters with no "=" padding,

(2) the final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by two "=" padding characters, or

(3) the final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output will be three characters followed by one "=" padding character.

3. Base 32 Encoding

The following description of base 32 is due to [7] (with corrections) and [6] (the "extended hex" alphabet).

The Base 32 encoding is designed to represent arbitrary sequences of octets in a form that needs to be case insensitive but need not be

humanly readable.

A 33-character subset of US-ASCII is used, enabling 5 bits to be represented per printable character. (The extra 33rd character, "=", is used to signify a special processing function.)

The encoding process represents 40-bit groups of input bits as output strings of 8 encoded characters. Proceeding from left to right, a 40-bit input group is formed by concatenating 5 8bit input groups. These 40 bits are then treated as 8 concatenated 5-bit groups, each of which is translated into a single digit in the base 32 alphabet. When encoding a bit stream via the base 32 encoding, the bit stream must be presumed to be ordered with the most-significant-bit first. That is, the first bit in the stream will be the high-order bit in the first 8bit byte, and the eighth bit will be the low-order bit in the first 8bit byte, and so on.

Each 5-bit group is used as an index into an array of 32 printable characters. The character referenced by the index is placed in the output string. These characters, identified in Table 2, below, are selected from US-ASCII digits and uppercase letters.

Table 3: The "Canonical" Base 32 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
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	(pad)	=
7	H	16	Q	25	Z		
8	I	17	R	26	2		

Table 4: The "Extended Hex" Base 32 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
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	(pad)	=
7	7	16	G	25	P		
8	8	17	H	26	Q		

Special processing is performed if fewer than 40 bits are available at the end of the data being encoded. A full encoding quantum is always completed at the end of a body. When fewer than 40 input bits are available in an input group, zero bits are added (on the right) to form an integral number of 5-bit groups. Padding at the end of the data is performed using the "=" character. Since all base 32 input is an integral number of octets, only the following cases can arise:

(1) the final quantum of encoding input is an integral multiple of 40 bits; here, the final unit of encoded output will be an integral multiple of 8 characters with no "=" padding,

(2) the final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by six "=" padding characters,

(3) the final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output will be four characters followed by four "=" padding characters,

(4) the final quantum of encoding input is exactly 24 bits; here, the final unit of encoded output will be five characters followed by three "=" padding characters, or

(5) the final quantum of encoding input is exactly 32 bits; here, the final unit of encoded output will be seven characters followed by one "=" padding character.

4. Base 16 Encoding

The following description is original but analogous to previous descriptions.

A 16-character subset of US-ASCII is used, enabling 4 bits to be represented per printable character.

The encoding process represents 8-bit groups (octets) of input bits as output strings of 2 encoded characters. Proceeding from left to right, a 8-bit input is taken from the input data. These 8 bits are then treated as 2 concatenated 4-bit groups, each of which is translated into a single digit in the base 16 alphabet.

Each 4-bit group is used as an index into an array of 16 printable characters. The character referenced by the index is placed in the output string.

This draft describe two alphabets for Base 16 encoding. While the Hex alphabet is arguable more natural there may be situations with special constraints, such as forbidden leading digits in strings, which the other may be useful. Both alphabets are to be handled case insensitive.

Table 5: The "Hex" Base 16 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
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

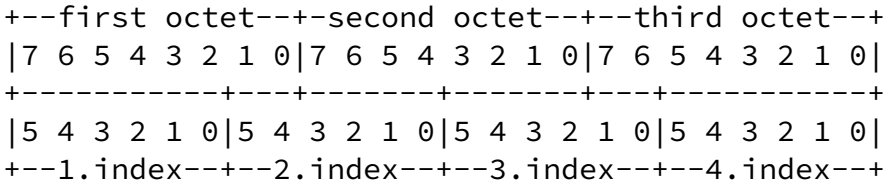
Table 6: The Canonical Base 16 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	4	E	8	I	12	M
1	B	5	F	9	J	13	N
2	C	6	G	10	K	14	O
3	D	7	H	11	L	15	P

Unlike base 32 and base 64, no special padding is necessery since a full code word is always available.

5. Examples

To translate between binary and a base encoding, the input is stored in a structure and the output is extracted. The case for base 64 is displayed in the following figure, borrowed from [4].



5.1 Examples of Base 64

This example is from [4].

```
Input data:  0x14fb9c03d97e
Hex:     1   4    f   b    9   c     | 0   3    d   9    7   e
8-bit:   00010100 11111011 10011100  | 00000011 11011001
11111110
6-bit:   000101 001111 101110 011100 | 000000 111101 100111
111110
Decimal: 5      15     46     28       0      61     37     62
Output:  F      P      u      c        A      9      l      +

Input data:  0x14fb9c03d9
Hex:     1   4    f   b    9   c     | 0   3    d   9
8-bit:   00010100 11111011 10011100  | 00000011 11011001
                                              pad with 00
6-bit:   000101 001111 101110 011100 | 000000 111101 100100
Decimal: 5      15     46     28       0      61     36
                                                  pad with =
Output:  F      P      u      c        A      9      k      =

Input data:  0x14fb9c03
```

```
Hex:     1   4    f   b    9   c     | 0   3
8-bit:   00010100 11111011 10011100  | 00000011
                                       pad with 0000
6-bit:   000101 001111 101110 011100 | 000000 110000
Decimal: 5      15     46     28       0      48
                                            pad with =      =
Output:  F      P      u      c        A      w      =      =
```


6. Security Considerations

When implementing Base 64 encoding and decoding, care should be made not to introduce vulnerabilities to buffer overflows.

7. Acknowledgement

I'd like to thank Tony Hansen and Gordon Mohr for comments and suggestions.

References

[1] Linn, J., "Privacy enhancement for Internet electronic mail: Part I - message encipherment and authentication procedures", RFC 1113, August 1989.

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

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

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

[5] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

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

[7] Myers, J., "SASL GSSAPI mechanisms", draft draft-ietf-cat-sasl-gssapi-01, May 2000.

[8] Zooko, O., "Post to P2P-hackers mailing list", World Wide Web http://zgp.org/pipermail/p2p-hackers/2001-September/000315.html, September 2001.

Author's Address

Simon Josefsson
Drottningholmsv. 70
Stockholm 112 42
Sweden

EMail: simon@josefsson.org

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.