Independent
Internet-Draft
Intended status: Standards Track
Expires: 12 October 2026

D. Crockford

K. R. Davis
Cisco Systems
10 April 2026

Base32 for Humans
draft-crockford-davis-base32-for-humans-01

Abstract

This document details the formal specification of Base32 for Humans; an alternate Base32 alphabet which extends the "Handled by Humans" text found in Section 3.4 of RFC 4648. This base features an alphabet curated for expressing numbers in a form that can be conveniently and accurately transmitted between humans and computer systems.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://github.com/douglascrockford/Base32/blob/draft-00/draft-crockford-davis-base32-for-humans.md. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-crockford-davis-base32-for-humans/.

Discussion of this document takes place on the Revise Universally Unique Identifier Definitions (uuidrev) Working Group mailing list (mailto:uuidrev@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/uuidrev/. Subscribe at https://www.ietf.org/mailman/listinfo/uuidrev/.

Source for this draft and an issue tracker can be found at https://github.com/douglascrockford/Base32.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

This Internet-Draft will expire on 12 October 2026.

Copyright Notice

Table of Contents

1. Introduction

Douglas Crockford’s Base32 has been published at [DCB32] since the early 2000s. This Base32 variant alphabet has been widely implemented and tested across the industry as an alternative Base32 ([RFC4648], Section 6) and Base32hex ([RFC4648], Section 7); however it does not have a standards-based document published with any standards body. Since Base32 itself was published with the IETF; it is fitting to publish this Base32 variant with the IETF as well.

This document serves to provide a standards-based reference for new libraries and for future RFCs to cite. One example is its use with Universally Unique IDentifiers (UUIDs) defined in [RFC9562], where Base32 for Humans is a recommended alternate encoding method via [new-uuid-encoding-techniques-ietf-draft].

Where possible, the content of this document mirrors the text of the original webpage, with expanded commentary and clarification where required, such as new encoding and decoding test vectors and comparisons to newer BaseXX alphabets.

1.1. Goals

The goals of this Base32 alphabet are as follows:

* Be human readable and machine readable.
* Be compact: Humans have difficulty in manipulating long strings of arbitrary symbols.
* Be error resistant: Entering the symbols must not require keyboarding gymnastics.
* Be pronounceable: Humans should be able to accurately transmit the symbols to other humans using a telephone.

2. Comparing Existing Alphabets

When examining other BaseXX alphabets against the goals defined in Section 1.1, the following observations are true:

* Base10 and Base16 ([RFC4648], Section 8) are well-known alphabets but tend to produce encoded outputs that are too long for human consumption.
* Base32 ([RFC4648], Section 6) and Base32hex ([RFC4648], Section 7) strike a balance between compactness and error resistance but include characters that are challenging for human use cases (see [RFC4648], Section 3.4).
* Base36 often uses the entire alphanumeric range and cannot easily substitute problematic characters.

* Base58 used in Bitcoin ([Base58btc]) removes similar problematic characters (0, Ii, Oo, Ll) but introduces both uppercase and lowercase letters, which can cause issues for case-sensitive inputs.
* Base62 ([IEEEBase62]) is a large alphabet with no symbols but, like Base36, uses uppercase and lowercase letters and therefore shares the same issues as Base36 and Base58.
* Base64 ([RFC4648], Section 4) and Base64url ([RFC4648], Section 5) provide compact encodings but require uppercase, lowercase, and special characters; removing any alphanumeric character typically forces a special character into the alphabet, which is undesirable for human-oriented use.

The length of the Base32 alphabet (32 characters) allows selecting a subset of characters while omitting problematic ones. While this is not the most compact encoding available, avoiding symbols and treating alphabet characters as equivalent improves human readability and reduces transcription errors when reading aloud or typing.

Compared to Z-Base-32 ([ZB32]), many design considerations overlap, but Z-Base-32 removes a different set of characters (0, 2, Ll, Vv). Z-Base-32 also reorders the alphabet in order to make certain characters appear more frequently for its target use cases. As a result, Z-Base-32 does not follow US-ASCII collation ([RFC20]) and does not sort the same as its binary counterpart.

3. The Alphabet

The Base32 for Humans alphabet, which can be referenced as "Base32plus", is a superset of the Base16 alphabet and an alternate version of the Base32hex alphabet; both featuring US-ASCII [RFC20] characters.

The encoded data conveys each character as a 5-bit value. If necessary, values are zero-extended as described in Section 3.1 so the input data length is a multiple of 5 bits.

Base32plus excludes four letters: I, L, O, and U.

I/i and l/L can be confused with the number 1, O/o can be confused with 0, and U can produce accidental obscenities or be confused with V/v. Note that 5/S/s and 2/Z/z are not modified in Base32plus, although these characters can look similar in handwriting (see [ZB32]).

Table 1 details the alphabet characters along with their respective decoding and encoding values. When encoding, only uppercase letters are used. For more information on decoding, see Section 4.

Value	Decoding	Encoding
00	0 O o	0
01	1 I i L l	1
02	2	2
03	3	3
04	4	4
05	5	5
06	6	6
07	7	7
08	8	8
09	9	9
10	A a	A
11	B b	B
12	C c	C
13	D d	D
14	E e	E
15	F f	F
16	G g	G
17	H h	H
18	J j	J
19	K k	K
20	M m	M
21	N n	N
22	P p	P

23	Q q	Q
24	R r	R
25	S s	S
26	T t	T
27	V v	V
28	W w	W
30	X x	X
31	Y y	Y
32	Z z	Z

Table 1: The Base32 for Humans Alphabet as table

The Alphabet as a continuous text input can be found in Figure 1.

```
0123456789ABCDEFGHJKMNPQRSTVWXYZ
```

Figure 1: The Base32 for Humans Alphabet as text

3.1. Padding

The Base32plus alphabet does not use a special padding character.

If the bit length of the input is not a multiple of 5, zero-extend the number in the least-significant bit positions to make the length a multiple of 5.

For example if the data is a 4-bit value 0b1111 the padded data would be a 5-bit value 0b11110.

3.2. Checksums

An application MAY append a check symbol to a symbol string. This check symbol can detect symbol-substitution and symbol-transposition errors, allowing transmission and entry errors to be caught cheaply and early.

The check symbol encodes the number modulo 37, the smallest prime greater than 32. Five additional symbols are defined for encoding or decoding the check symbol; these are shown in The Base32 for Humans Checksum Values and extend The Base32 for Humans Alphabet as table.

These additional symbols were chosen to avoid confusion with punctuation or URL formatting.

Value	Decoding	Encoding
32	*	*
33	~	~
34	$	$
35	=	=
36	U u	U

Table 2: The Base32 for Humans Checksum Values

The the checksums postfixed to the The Base32 for Humans Alphabet as text can be found in The Base32 for Humans Alphabet with Checksums.

0123456789ABCDEFGHJKMNPQRSTVWXYZ*~$=U

Figure 2: The Base32 for Humans Alphabet with Checksums

Where these symbols prove problematic for various application use cases, implementations MAY strip checksum values, utilize percent-encoding, or other application-specific methods to ensure proper handling and delivery of the checksum value.

4. Decoding

When decoding, both uppercase and lowercase letters are accepted. I/i and L/l are treated as 1, and O/o is treated as 0.

Hyphens (-) MAY be inserted into symbol strings to partition them into manageable pieces, improving readability and reducing confusion. Hyphens are ignored during decoding, although an application MAY check hyphen placement to help validate a symbol string.

5. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

6. Security Considerations

Section Checksums addresses the primary security-related concern in this document: data integrity and validation of decoded data.

No additional security considerations are identified.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[RFC20] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/rfc/rfc20>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

8.2. Informative References

[Base58btc] Bitcoin, "Base58 Bitcoin", commit fae71d3, November 2008, <https://github.com/bitcoin/bitcoin/blob/master/src/base58.cpp>.

[DCB32] Crockford, D., "Douglas Crockford's Base32", n.d., <https://www.crockford.com/base32.html>.

[IEEEBase62] IEEE, "A secure, lossless, and compressed Base62 encoding", November 2008, <https://ieeexplore.ieee.org/document/4737287>.

[new-uuid-encoding-techniques-ietf-draft] Davis, K., "Alternate UUID Encoding Methods", n.d., <https://datatracker.ietf.org/doc/draft-davis-uuidrev-alt-uuid-encoding-methods/>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/rfc/rfc9562>.

[ZB32] O'Whielacronx, Z., "human-oriented base-32 encoding", November 2009, <https://philzimmermann.com/docs/human-oriented-base-32-encoding.txt>.

Appendix A. Changelog

draft-00:

* Initial Release

draft-02:

* update front matter venue and alt encoding reference

Appendix B. Test Vectors

The following text vectors start with a base of those found in [RFC4648] and then add additional values that illustrate decoding of specific values and the usage of the checksum.

B.1. Encoding

Input	Encoded Data	Checksum?	Input Type
f	CR	N	Text
fo	CSQG	N	Text
foo	CSQPY	N	Text

foob	CSQPYRG	N	Text
fooba	CSQPYRK1	N	Text
foobar	CSQPYRK1E8	N	Text
foobar	CSQPYRK1E86	Y	Text
test	EHJQ6X0	N	Text
test	EHJQ6X0V	Y	Text
123456789	0XDWT58U	Y	Integer

Table 3: Base32 for Humans Encode Table

B.2. Decoding

Input	Decoded Data	Test Info
CSQPYRK1E8	foobar	Basic Input 1, decoded as text
CSQPY-RK1E8	foobar	Decode as text with dashes
csqpyrkle8	foobar	Decode l input as text
csqpyrkie8	foobar	Decode i input as text
EHJQ6X0	test	Base Input 2, decoded as text
ehjq6xo	test	Decode o input as text
CSQPYRK1E86	foobar	Checksum 1, decoded as text
EHJQ6X0V	test	Checksum 2, decoded as text
0XDWT58U	123456789	Checksum value of extended alphabet, decoded as integer

Table 4: Base32 for Humans Decode Table

Authors' Addresses

Douglas Crockford
Email: douglas@crockford.com

Kyzer R. Davis
Cisco Systems
Email: kydavis@cisco.com, kyzer.davis@outlook.com