Workgroup: Network Working Group
Internet-Draft: draft-multiformats-multibase-05
Published: February 2022
Intended Status: Informational
Expires: 20 August 2022
J. Benet
Protocol Labs
M. Sporny
Digital Bazaar

The Multibase Data Format

Abstract

Raw binary data is often encoded using a mechanism that enables the data to be included in human-readable text-based formats. This mechanism is often referred to as "base-encoding the data". Base-encoding is often used when expressing binary data in hyperlinks, cryptographic keys in web pages, or security tokens in application software. There are a variety of base-encodings, such as base32, base58, and base64. It is not always possible to differentiate one base-encoding from another. The purpose of this specification is to provide a mechanism to be able to deterministically identify the base-encoding for a particular string of data.

Feedback

This specification is a joint work product of Protocol Labs, the W3C Digital Verification Community Group, and the W3C Credentials Community Group. Feedback related to this specification should logged in the issue tracker or be sent to public-credentials@w3.org. .

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 August 2022.

Copyright Notice

Table of Contents

1. Introduction

This specification describes a forward-compatible data model for expressing raw binary data in a variety of base-encoding formats such as base32, base58. and base64.

When text is encoded as bytes, we can usually use a one-size-fits-all encoding (UTF-8) because we're always encoding to the same set of 256 bytes. When that doesn't work, usually for historical or performance reasons, we can usually infer the encoding from the context.

However, when bytes are encoded as text (using a base encoding), the choice of base encoding is often restricted by the context. Worse, these restrictions can change based on where the data appears in the text. In some cases, we can only use [a-z0-9]. In others, we can use a larger set of characters but need a compact encoding. This has lead to a large set of "base encodings", one for every use-case. Unlike when encoding text to bytes, we can't just standardize around a single base encoding because there is no optimal encoding for all cases.

Unfortunately, it's not always clear what base encoding is used; that's where this specification comes in. It answers the question:

Given data 'd' encoded into text 's', what base is it encoded with?

2. The Multibase Format

A multibase-encoded value follows a simple format:

base-encoding-character base-encoded-data

The encoding algorithm is a single character value that is always the first byte of the data. The possible values for this field are provided in The Multibase Algorithm Registry.

2.1. A Multibase Example

The following is an encoding of "Hello World!" using the version of base-58 that utilizes the Bitcoin encoding character set:

```
z2NEpo7TZRRrLZSi2U
```

The first byte (z) specifies the multibase encoding algorithm. The rest of the data specifies the value of the output of the multibase encoding algorithm.

3. Normative References

[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/info/rfc2119>.

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

Appendix A. Security Considerations

There are a number of security considerations to take into account when implementing or utilizing this specification. TBD

Appendix B. Test Values

The multibase examples are chosen to show different encoding algorithms and different output lengths at play. The input test data for all of the examples in this section is:

```
Multibase is awesome! \o/
```

B.1. Hexadecimal upper-case encoding

```
F4D756C7469626173652069732061765736F6D6521205C6F2F
```

B.2. Base-32 upper-case encoding, no padding

```
BJV2WY5DJMJQXGZJANFZSAYLXMVZW63LFEEQFY3ZP
```

B.3. Base-58 Bitcoin encoding

```
zYAjKoNbau5KiqmHPmSxYCvn66dA1vLmwbt
```

B.4. Base-64 with padding and MIME-encoding

```
MTXVsdGliYXNlIGlzIGF3ZXNvbWUhIFxvLw==
```

Appendix C. Acknowledgements

The editors would like to thank the following individuals for feedback on and implementations of the specification (in alphabetical order):

Appendix D. IANA Considerations

D.1. The Multibase Algorithms Registry

The following initial entries should be added to the Multibase Algorithms Registry to be created and maintained at (the suggested URI) http://www.iana.org/assignments/multibase-algorithms:

Algorithm	Identifier (character)	Status	Specification
identity	0x00	active	8-bit binary (encoder and decoder keeps data unmodified)
base2	0	active	binary (01010101)
base8	7	active	octal
base10	9	active	decimal
base16	f	active	hexadecimal
base16upper	F	active	hexadecimal
base32hex	v	active	RFC 4648 [RFC4648] case-insensitive - no padding - highest char
base32hexupper	V	active	RFC 4648 [RFC4648] case-insensitive - no padding - highest char
base32hexpad	t	active	RFC 4648 [RFC4648] case-insensitive - with padding
base32hexpadupper	T	active	RFC 4648 [RFC4648] case-insensitive - with padding
base32	b	active	RFC 4648 [RFC4648] case-insensitive - no padding
base32upper	B	active	RFC 4648 [RFC4648] case-insensitive - no padding
base32pad	c	active	RFC 4648 [RFC4648] case-insensitive - with padding
base32padupper	C	active	RFC 4648 [RFC4648] case-insensitive - with padding
base32z	h	active	z-base-32 (used by Tahoe-LAFS)
base36	k	active	base36 [0-9a-z] case-insensitive - no padding
base36upper	K	active	base36 [0-9a-z] case-insensitive - no padding
base58btc	z	active	base58 bitcoin
base58flickr	Z	active	base58 flicker
base64	m	active	RFC 4648 [RFC4648] no padding
base64pad	M	active	RFC 4648 [RFC4648] with padding - MIME encoding

Algorithm	Identifier (character)	Status	Specification
base64url	u	active	RFC 4648 [RFC4648] no padding
base64urlpad	U	active	RFC 4648 [RFC4648] with padding
proquint	p	active	PRO-QUINT https://arxiv.org/html/0901.4016

Table 1: Multihash Algorithms Registry

NOTE: The most up to date place for developers to find the table above is https://github.com/multiformats/multibase/blob/master/multibase.csv.

Authors' Addresses

Juan Benet
Protocol Labs
548 Market Street, #51207
San Francisco, CA 94104
United States of America

Phone: +1 619 957 7606
Email: juan@protocol.ai
URI: http://juan.benet.ai/

Manu Sporny
Digital Bazaar
203 Roanoke Street W.
Blacksburg, VA 24060
United States of America

Phone: +1 540 961 4469
Email: msporny@digitalbazaar.com
URI: http://manu.sporny.org/