The Multibase Data Format
draft-multiformats-multibase-02

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 [1], the
   W3C Digital Verification Community Group [2], and the W3C Credentials
   Community Group [3]. Feedback related to this specification should
   logged in the issue tracker [4] or be sent to public-
   credentials@w3.org [5].   .

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 February 19, 2021.

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 [6].

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. References

3.1. 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>.

3.2. URIs

[1] https://protocol.ai/

[2] https://w3c-dvcg.github.io/

   [3] https://w3c-ccg.github.io/

   [4] https://github.com/w3c-dvcg/multibase/issues

   [5] mailto:public-credentials@w3.org

   [6] #mb-registry

   [7] http://www.iana.org/assignments/multibase-algorithms

   [8] https://github.com/multiformats/multibase/blob/master/
        multibase.csv

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

   F4D756C74696261736520697320617765736F6D6521205C6F2F

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 [7]:

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
base64url	u	active	RFC 4648 [RFC4648] no padding
base64urlpad	U	active	RFC 4648 [RFC4648] with padding

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 [8].

Authors' Addresses

Juan Benet
Protocol Labs
548 Market Street, #51207
San Francisco, CA 94104
US

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
US

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