Workgroup: Network Working Group
Internet-Draft:
draft-multiformats-multihash-07
Published: 20 August 2023
Intended Status: Informational
Expires: 21 February 2024
Authors: J. Benet
Protocol Labs
M. Sporny
Digital Bazaar

The Multihash Data Format

Abstract

Cryptographic hash functions often have multiple output sizes and encodings. This variability makes it difficult for applications to examine a series of bytes and determine which hash function produced them. Multihash is a universal data format for encoding outputs from hash functions. It is useful to write applications that can simultaneously support different hash function outputs as well as upgrade their use of hashes over time; Multihash is intended to address these needs.

Feedback

This specification is a joint work product of Protocol Labs 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 21 February 2024.

Table of Contents

1. Introduction

Multihash is particularly important in systems which depend on cryptographically secure hash functions. Attacks may break the cryptographic properties of secure hash functions. These cryptographic breaks are particularly painful in large tool ecosystems, where tools may have made assumptions about hash values, such as function and digest size. Upgrading becomes a nightmare, as

all tools which make those assumptions would have to be upgraded to use the new hash function and new hash digest length. Tools may face serious interoperability problems or error-prone special casing.

How many programs out there assume a git hash is a SHA-1 hash?

How many scripts assume the hash value digest is exactly 160 bits?

How many tools will break when these values change?

How many programs will fail silently when these values change?

This is precisely why Multihash was created. It was designed for seamlessly upgrading systems that depend on cryptographic hashes.

When using Multihash, a system warns the consumers of its hash values that these may have to be upgraded in case of a break. Even though the system may still only use a single hash function at a time, the use of multihash makes it clear to applications that hash values may use different hash functions or be longer in the future. Tooling, applications, and scripts can avoid making assumptions about the length, and read it from the multihash value instead. This way, the vast majority of tooling - which may not do any checking of hashes - would not have to be upgraded at all. This vastly simplifies the upgrade process, avoiding the waste of hundreds or thousands of software engineering hours, deep frustrations, and high blood pressure.

2. The Multihash Fields

A multihash follows the TLV (type-length-value) pattern and consists of several fields composed of a combination of unsigned variable length integers and byte information.

2.1. Multihash Core Data Types

The following section details the core data types used by the Multihash data format.

2.1.1. unsigned variable integer

A data type that enables one to express an unsigned integer of variable length. The format uses the Little Endian Base 128 (LEB128) encoding that is defined in Appendix C of the DWARF Debugging Information Format [DWARF] standard, initially released in 1993.

As suggested by the name, this variable length encoding is only capable of representing unsigned integers. Further, while there is no theoretical maximum integer value that can be represented by the format, implementations MUST NOT encode more than nine (9) bytes

giving a practical limit of integers in a range between 0 and 2^63 - 1.

When encoding an unsigned variable integer, the unsigned integer is serialized seven bits at a time, starting with the least significant bits. The most significant bit in each output byte indicates if there is a continuation byte. It is not possible to express a signed integer with this data type.

Value	Encoding (bits)	hexadecimal notation
1	00000001	0x01
127	01111111	0x7F
128	10000000 00000001	0x8001
255	11111111 00000001	0xFF01
300	10101100 00000010	0xAC02
16384	10000000 10000000 00000001	0x808001

Table 1: Examples of Unsigned Variable Integers

Implementations MUST restrict the size of the varint to a max of nine bytes (63 bits). In order to avoid memory attacks on the encoding, the aforementioned practical maximum length of nine bytes is used. There is no theoretical limit, and future specs can grow this number if it is truly necessary to have code or length values larger than 2^31.

2.2. Multihash Fields

A multihash follows the TLV (type-length-value) pattern.

2.2.1. Hash Function Identifier

The hash function identifier is an unsigned variable integer identifying the hash function. The possible values for this field are provided in The Multihash Identifier Registry.

2.2.2. Digest Length

The digest length is an unsigned variable integer counting the length of the digest in bytes.

2.2.3. Digest Value

The digest value is the hash function digest with a length of exactly what is specified in the digest length, which is specified in bytes.

2.3. A Multihash Example

For example, the following is an expression of a SHA2-256 hash in hexadecimal notation (spaces added for readability purposes):

```
0x12 20 41dd7b6443542e75701aa98a0c235951a28a0d851b11564d20022ab11d2589a8
```

The first byte (0x12) specifies the SHA2-256 hash function. The second byte (0x20) specifies the length of the hash, which is 32 bytes. The rest of the data specifies the value of the output of the hash function.

3. References

3.1. Normative References

[DWARF] Workgroup, D. D. I. F., Ed., "DWARF Debugging Information Format, Version 3", December 2005, <http://dwarfstd.org/doc/Dwarf3.pdf>.

[FIPS202] Technology, I. T. L. N. I. O. S. A., Ed., "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", FIPS 202, DOI 10.6028/NIST.FIPS.202, August 2015, <https://doi.org/10.6028/NIST.FIPS.202>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC7693] Saarinen, M., Ed. and J. Aumasson, "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)", RFC 7693, DOI 10.17487/RFC7693, November 2015, <https://www.rfc-editor.org/info/rfc7693>.

3.2. Informative References

[RFC6150] Turner, S. and L. Chen, "MD4 to Historic Status", RFC 6150, DOI 10.17487/RFC6150, March 2011, <https://www.rfc-editor.org/info/rfc6150>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

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 multihash examples are chosen to show different hash functions and different hash digest lengths at play. The input test data for all of the examples in this section is:

Merkle–Damgård

B.1. SHA-1

0x11148a173fd3e32c0fa78b90fe42d305f202244e2739

The fields for this multihash are - hashing function: sha1 (0x11), length: 20 (0x14), digest: 0x8a173fd3e32c0fa78b90fe42d305f202244e2739

B.2. SHA-256

0x122041dd7b6443542e75701aa98a0c235951a28a0d851b11564d20022ab11d2589a8

The fields for this multihash are - hashing function: sha2-256 (0x12), length: 32 (0x20), digest: 0x41dd7b6443542e75701aa98a0c235951a28a0d851b11564d20022ab11d2589a8

B.3. SHA-512/256

0x132052eb4dd19f1ec522859e12d89706156570f8fbab1824870bc6f8c7d235eef5f4

The fields for this multihash are - hashing function: sha2-512 (0x13), length: 32 (0x20), digest: 0x52eb4dd19f1ec522859e12d89706156570f8fbab1824870bc6f8c7d235eef5f4

B.4. SHA-512

0x134052eb4dd19f1ec522859e12d89706156570f8fbab1824870bc6f8c7d235eef5f4c2

The fields for this multihash are - hashing function: sha2-512 (0x13), length: 64 (0x40), digest: 0x52eb4dd19f1ec522859e12d89706156570f8fbab1824870bc6f8c7d235eef5f4c2cbbafd365f96fb12b1d98a0334870c2ce90355da25e6a1108a6e17c4aaebb0

B.5. blake2b512

0xb24040d91ae0cb0e48022053ab0f8f0dc78d28593d0f1c13ae39c9b169c136a779f21a

The fields for this multihash are - hashing function: blake2b-512 (0xb240), length: 64 (0x40), digest: 0xd91ae0cb0e48022053ab0f8f0dc78d28593d0f1c13ae39c9b169c136a779f21a0496337b6f776a73c1742805c1cc15e792ddb3c92ee1fe300389456ef3dc97e2

B.6. blake2b256

0xb220207d0a1371550f3306532ff44520b649f8be05b72674e46fc24468ff74323ab030

The fields for this multihash are - hashing function: blake2b-256 (0xb220), length: 32 (0x20), digest: 0x7d0a1371550f3306532ff44520b649f8be05b72674e46fc24468ff74323ab030

B.7. blake2s256

0xb26020a96953281f3fd944a3206219fad61a40b992611b7580f1fa091935db3f7ca13d

The fields for this multihash are - hashing function: blake2s-256 (0xb260), length: 32 (0x20), digest: 0xa96953281f3fd944a3206219fad61a40b992611b7580f1fa091935db3f7ca13d

B.8. blake2s128

0xb250100a4ec6f1629e49262d7093e2f82a3278

The fields for this multihash are - hashing function: blake2s-128 (0xb250), length: 16 (0x10), digest: 0x0a4ec6f1629e49262d7093e2f82a3278

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 Multihash Identifier Registry

The Multihash Identifier Registry contains hash functions supported by Multihash each with its canonical name, its value in hexadecimal notation, and its status. The following initial entries should be added to the registry to be created and maintained at (the suggested URI) http://www.iana.org/assignments/multihash-identifiers:

Name	Identifier	Status	Specification
identity	0x00	active	Unknown
sha1	0x11	active	RFC 6234 [RFC6234]
sha2-256	0x12	active	RFC 6234 [RFC6234]
sha2-512	0x13	active	RFC 6234 [RFC6234]
sha3-512	0x14	active	FIPS 202 [FIPS202]
sha3-384	0x15	active	FIPS 202 [FIPS202]
sha3-256	0x16	active	FIPS 202 [FIPS202]
sha3-224	0x17	active	FIPS 202 [FIPS202]

Name	Identifier	Status	Specification
sha2-384	0x20	active	RFC 6234 [RFC6234]
sha2-256-trunc254-padded	0x1012	active	RFC 6234 [RFC6234]
sha2-224	0x1013	active	RFC 6234 [RFC6234]
sha2-512-224	0x1014	active	RFC 6234 [RFC6234]
sha2-512-256	0x1015	active	RFC 6234 [RFC6234]
blake2b-256	0xb220	active	RFC 7693 [RFC7693]
poseidon-bls12_381-a2-fc1	0xb401	active	Unknown

Table 2: Multihash Identifier Registry

NOTE: The most up to date place for developers to find the table above, plus all multihash headers in "draft" status, is https://github.com/multiformats/multicodec/blob/master/table.csv.

D.2. The 'mh' Digest Algorithm

This memo registers the "mh" digest-algorithm in the HTTP Digest Algorithm Values registry with the following values:

Digest Algorithm: mh

Description: The multibase-serialized value of a multihash-supported algorithm.

References: this document

Status: standard

D.3. The 'mh' Named Information Hash Algorithm

This memo registers the "mh" hash algorithm in the Named Information Hash Algorithm registry with the following values:

ID: 49

Hash Name String: mh

Value Length: variable

Reference: this document

Status: current

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/