HTTP
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2020

R. Polli
Team Digitale, Italian Government
L. Pardue
Cloudflare

July 04, 2019

Resource Digests for HTTP
draft-ietf-httpbis-digest-headers-00

Abstract

This document defines the Digest and Want-Digest header fields for HTTP, thus allowing client and server to negotiate an integrity checksum of the exchanged resource representation data.

This document obsoletes RFC 3230. It replaces the term "instance" with "representation", which makes it consistent with the HTTP Semantic and Context defined in RFC 7231.

Note to Readers

RFC EDITOR: please remove this section before publication

Discussion of this draft takes place on the HTTP working group mailing list (ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/ [1].

The source code and issues list for this draft can be found at https://github.com/httpwg/http-extensions [2].

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 January 5, 2020.

Table of Contents

1. Introduction

Integrity protection for HTTP content is multi layered and is usually achieved across the protocol stack: TCP checksums and TLS [RFC2818] record to name but some.

The HTTP protocol does not provide means to protect the various message parts. Besides, it might be desirable to add additional guarantees to the ones provided by the transport layer (eg. HTTPS). This may be in order to:

- detect programming errors and corruption of stored data;
- address the need for the representation-data to remain unmodified throughout multiple hops;
- implement signature mechanisms that cover the desired parts of an HTTP exchange;
- provide additional protection against failures or attack (see [SRI]).

1.1. Brief history of integrity headers

The Content-MD5 header field was originally introduced to provide integrity, but HTTP/1.1 [RFC7231] in appendix-B obsoleted it:

> The Content-MD5 header field has been removed because it was inconsistently implemented with respect to partial responses.

[RFC3230] provided a more flexible solution introducing the concept of "instance", and the headers "Digest" and "Want-Digest".

1.2. This proposal

The concept of "selected representation" defined in [RFC7231] made [RFC3230] definitions inconsistent with the current standard. A refresh was then required.

This document updates the "Digest" and "Want-Digest" header field definitions to align with [RFC7231] concepts.

This approach can be easily adapted to use-cases where the transferred data does require some sort of manipulation to be considered a representation or conveys a partial representation of a resource (eg. Range Requests [RFC7233]).

Changes are semantically compatible with existing implementations and better cover both the request and response cases.

The value of "Digest" is calculated on selected representation, which is tied to the value contained in any "Content-Encoding" or "Content-Type" header fields. Therefore, a given resource may have multiple different digest values.

To allow both parties to exchange a Digest of a representation with no content codings [3] two more algorithms are added ("ID-SHA-256" and "ID-SHA-512").

1.3. Goals

The goals of this proposal are:

1. Digest coverage for either the resource's "representation data" or "selected representation data" communicated via HTTP.

2. Support for multiple digest algorithms.

3. Negotiation of the use of digests.

The goals do not include:

HTTP Message integrity: The digest mechanism described here does not cover the full HTTP message nor its semantic, as representation metadata are not included in the checksum.

Header integrity: The digest mechanisms described here cover only representation and selected representation data, and do not protect the integrity of associated representation metadata headers or other message headers.

Authentication: The digest mechanisms described here are not meant to support authentication of the source of a digest or of a message or anything else. These mechanisms, therefore, are not a sufficient defense against many kinds of malicious attacks.

Privacy: Digest mechanisms do not provide message privacy.

Authorization: The digest mechanisms described here are not meant to support authorization or other kinds of access controls.

1.4. Notational Conventions

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] and [RFC8174]) when, and only when, they appear in all capitals, as shown here.

This document uses the Augmented BNF defined in [RFC5234] and updated by [RFC7405] along with the "#rule" extension defined in Section 7 of [RFC7230].

The definitions "representation", "selected representation", "representation data", "representation metadata" and "payload body" in this document are to be interpreted as described in [RFC7230] and [RFC7231].

2. Resource representation and representation-data

To avoid inconsistencies, an integrity mechanism for HTTP messages should decouple the checksum calculation:

- from the payload body - which may be altered by mechanism like Range Requests [RFC7233] or the method (eg. HEAD);
- and from the message body - which depends on "Transfer-Encoding" and whatever tranformations the intermediaries may apply.

The following examples show how representation metadata, payload tranformations and method impacts on the message and payload body.

Here is a gzip-compressed json object

Request:

```
PUT /entries/1234 HTTP/1.1
Content-Type: application/json
Content-Encoding: gzip

H4sIAItWyFwC/6tWSlSyUlAypANQqgUAREcqfG0AAAA=
```

Now the same payload body conveys a malformed json object.

Request:

```
PUT /entries/1234 HTTP/1.1
Content-Type: application/json

H4sIAItWyFwC/6tWSlSyUlAypANQqgUAREcqfG0AAAA=
```

A Range-Request alters the payload body, conveying a partial representation.

Request:

```
GET /entries/1234 HTTP/1.1
Range: bytes=1-7
```

Response:

```
HTTP/1.1 206 Partial Content
Content-Encoding: gzip
Content-Type: application/json
Content-Range: bytes=1-7

iwgAla3RXA==
```

Now the method too alters the payload body.

Request:

```
HEAD /entries/1234 HTTP/1.1
Accept: application/json
Accept-Encoding: gzip
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Encoding: gzip
```

3. Digest Algorithm values

Digest algorithm values are used to indicate a specific digest computation. For some algorithms, one or more parameters may be supplied.

```
digest-algorithm = token
```

The BNF for "parameter" is as is used in [RFC7230]. All digest-algorithm values are case-insensitive.

The Internet Assigned Numbers Authority (IANA) acts as a registry for digest-algorithm values. The registry contains the following tokens.

SHA-256:

* Description: The SHA-256 algorithm [FIPS180-3]. The output of this algorithm is encoded using the base64 encoding [RFC4648].

* Reference: [FIPS180-3], [RFC4648], this document.

* Status: standard

SHA-512:

* Description: The SHA-512 algorithm [FIPS180-3]. The output of this algorithm is encoded using the base64 encoding [RFC4648].

* Reference: [FIPS180-3], [RFC4648], this document.

* Status: standard

MD5:

* Description: The MD5 algorithm, as specified in [RFC1321]. The output of this algorithm is encoded using the base64 encoding [RFC4648]. The MD5 algorithm is NOT RECOMMENDED as it's now vulnerable to collision attacks [CMU-836068].

* Reference: [RFC1321], [RFC4648], this document.

* Status: obsoleted

SHA:

* Description: The SHA-1 algorithm [FIPS180-1]. The output of this algorithm is encoded using the base64 encoding [RFC4648]. The SHA algorithm is NOT RECOMMENDED as it's now vulnerable to collision attacks [IACR-2019-459].

* Reference: [FIPS180-3], [RFC4648], this document.

* Status: obsoleted

UNIXsum:

* Description: The algorithm computed by the UNIX "sum" command, as defined by the Single UNIX Specification, Version 2 [UNIX]. The output of this algorithm is an ASCII decimal-digit string representing the 16-bit checksum, which is the first word of the output of the UNIX "sum" command.

* Reference: [UNIX], this document.

* Status: standard

UNIXcksum:

* Description: The algorithm computed by the UNIX "cksum" command, as defined by the Single UNIX Specification, Version 2 [UNIX]. The output of this algorithm is an ASCII digit string representing the 32-bit CRC, which is the first word of the output of the UNIX "cksum" command.

* Reference: [UNIX], this document.

* Status: standard

To allow sender and recipient to provide a checksum which is independent from "Content-Encoding", the following additional algorithms are defined:

ID-SHA-512:

* Description: The sha-512 digest of the representation-data of the resource when no content coding is applied (eg. "Content-Encoding: identity")

* Reference: [FIPS180-3], [RFC4648], this document.

* Status: standard

ID-SHA-256:

* Description: The sha-256 digest of the representation-data of the resource when no content coding is applied (eg. "Content-Encoding: identity")

* Reference: [FIPS180-3], [RFC4648], this document.

* Status: standard

If other digest-algorithm values are defined, the associated encoding MUST either be represented as a quoted string, or MUST NOT include ";" or "," in the character sets used for the encoding.

3.1. Representation digest

A representation digest is the value of the output of a digest algorithm, together with an indication of the algorithm used (and any parameters).

```
representation-data-digest = digest-algorithm "="
                          <encoded digest output>
```

As explained in Section 2 the digest is computed on the entire selected "representation data" of the resource defined in [RFC7231]:

```
representation-data := Content-Encoding( Content-Type( bits ) )
```

The encoded digest output uses the encoding format defined for the specific digest-algorithm.

3.1.1. digest-algorithm encoding examples

The "sha-256" digest-algorithm uses base64 encoding. Note that digest-algoritm values are case insensitive.

```
sha-256=X48E9qOokqqrvdts8nOJRJN3OWDUoyWxBf7kbu9DBPE=
```

The "UNIXsum" digest-algorithm uses ASCII string of decimal digits.

UNIXsum=30637

4. Header Specifications

The following headers are defined

4.1. Want-Digest

The Want-Digest message header field indicates the sender's desire to receive a representation digest on messages associated with the Request- URI and representation metadata.

```
Want-Digest = "Want-Digest" ":" OWS 1#want-digest-value
want-digest-value = digest-algorithm [ ";" "q" "=" qvalue]
qvalue = ( "0"  [ "."  0*1DIGIT ] ) /  ( "1"  [ "."  0*1( "0" ) ] )
```

If a digest-algorithm is not accompanied by a qvalue, it is treated as if its associated qvalue were 1.0.

The sender is willing to accept a digest-algorithm if and only if it is listed in a Want-Digest header field of a message, and its qvalue is non-zero.

If multiple acceptable digest-algorithm values are given, the sender's preferred digest-algorithm is the one (or ones) with the highest qvalue.

Examples:

```
Want-Digest: sha-256
Want-Digest: SHA-512;q=0.3, sha-256;q=1, md5;q=0
```

4.2. Digest

The Digest header field provides a digest of the representation data

```
Digest = "Digest" ":" OWS 1#representation-data-digest
```

"Representation data" might be:

- fully contained in the message body,
- partially-contained in the message body,
- or not at all contained in the message body.

The resource is specified by the effective Request-URI and any cache-validator contained in the message.

For example, in a response to a HEAD request, the digest is calculated using the representation data that would have been enclosed in the payload body if the same request had been a GET.

Digest can be used in requests too. Returned value depends on the representation metadata headers.

A Digest header field MAY contain multiple representation-data-digest values. This could be useful for responses expected to reside in caches shared by users with different browsers, for example.

A recipient MAY ignore any or all of the representation-data-digests in a Digest header field. This allows the recipient to chose which digest-algorithm(s) to use for validation instead of verifying every received representation-data-digest.

A sender MAY send a representation-data-digest using a digest-algorithm without knowing whether the recipient supports the digest-algorithm, or even knowing that the recipient will ignore it.

...

5. Deprecate Negotiation of Content-MD5

This RFC deprecates the negotiation of Content-MD5 as this header has been obsoleted by [RFC7231]

6. Broken cryptographic algorithms are NOT RECOMMENDED

The MD5 algorithm is NOT RECOMMENDED as it's now vulnerable to collision attacks [CMU-836068].

The SHA algorithm is NOT RECOMMENDED as it's now vulnerable to collision attacks [IACR-2019-459].

7. Examples

7.1. Unsolicited Digest response

7.1.1. Representation data is fully contained in the payload

Request:

```
GET /items/123
```

Response:

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Encoding: identity
Digest: sha-256=X48E9qOokqqrvdts8nOJRJN3OWDUoyWxBf7kbu9DBPE=

{"hello": "world"}
```

7.1.2. Representation data is not contained in the payload

Request:

```
HEAD /items/123
```

Response:

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Encoding: identity
Digest: sha-256=X48E9qOokqqrvdts8nOJRJN3OWDUoyWxBf7kbu9DBPE=
```

7.1.3. Representation data is partially contained in the payload i.e. range request

Request:

```
GET /items/123
Range: bytes=1-7
```

Response:

```
HTTP/1.1 206 Partial Content
Content-Type: application/json
Content-Encoding: identity
Content-Range: bytes 1-7/18
Digest: sha-256=X48E9qOokqqrvdts8nOJRJN3OWDUoyWxBf7kbu9DBPE=

"hello"
```

7.1.4. Digest in both Request and Response. Returned value depends on representation metadata

Digest can be used in requests too. Returned value depends on the representation metadata headers.

Request:

```
PUT /items/123
Content-Type: application/json
Content-Encoding: identity
Accept-Encoding: br
Digest: sha-256=4REjxQ4yrqUVicfSKYNO/cF9zNj5ANbzgDZt3/h3Qxo=

{"hello": "world"}
```

Response:

```
Content-Type: application/json
Content-Encoding: br
Digest: sha-256=X48E9qOokqqrvdts8nOJRJN3OWDUoyWxBf7kbu9DBPE=

iwiAeyJoZWxsbyI6ICJ3b3JsZCJ9Aw==
```

7.2. Want-Digest solicited digest responses

7.2.1. Client request data is fully contained in the payload

The client requests a digest, preferring sha. The server is free to reply with sha-256 anyway.

Request:

```
GET /items/123
Want-Digest: sha-256;q=0.3, sha;q=1
```

Response:

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Encoding: identity
Digest: sha-256=X48E9qOokqqrvdts8nOJRJN3OWDUoyWxBf7kbu9DBPE=

{"hello": "world"}
```

7.2.2. A client requests an unsupported Digest, the server MAY reply with an unsupported digest

The client requests a sha digest only. The server is currently free to reply with a Digest containing an unsupported algorithm

Request:

```
GET /items/123
Want-Digest: sha;q=1
```

Response:

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Encoding: identity
Digest: sha-256=X48E9qOokqqrvdts8nOJRJN3OWDUoyWxBf7kbu9DBPE=

{"hello": "world"}
```

7.2.3. A client requests an unsupported Digest, the server MAY reply with a 400

The client requests a sha Digest, the server advises for sha-256 and sha-512

Request:

```
GET /items/123
Want-Digest: sha;q=1
```

Response:

```
HTTP/1.1 400 Bad Request
Want-Digest: sha-256, sha-512
```

...

8. Security Considerations

8.1. Digest does not protect the full HTTP message

This document specifies a data integrity mechanism that protects HTTP "representation data", but not HTTP "representation metadata" headers, from certain kinds of accidental corruption.

While it is not intended as general protection against malicious tampering with HTTP messages, this goal can be achieved using

"Digest" together with a transport-layer security mechanism and digital signatures.

8.2. Broken cryptographic algorithms

Cryptogrphic alorithms are intended to provide a proof of integrity suited towards cryptographic constructions such as signatures.

However, these rely on collision-resistance for their security proofs [CMU-836068]. The MD5 and SHA-1 algorithms are vulnerable to collisions attacks and they are NOT RECOMMENDED.

8.3. Digest for end-to-end integrity

"Digest" alone does not provide end-to-end integrity of HTTP messages over multiple hops, as it just covers the "representation data" and not the "representation metadata".

Besides, it allows to protect "representation data" from buggy manipulation, buggy compression, etc.

Moreover identity digest algorithms (eg. ID-SHA-256 and ID-SHA-512) allow piecing together a resource from different sources (e.g. different servers that perhaps apply different content codings) enabling the user-agent to detect that the application-layer tasks completed properly, before handing off to say the HTML parser, video player etc.

Even a simple mechanism for end-to-end validation is thus valuable.

8.4. Usage in signatures

Digital signatures are widely used together with checksums to provide the certain identification of the origin of a message [NIST800-32].

It's important to note that, being the "Digest" header an hash of a resource representation, signing only the "Digest" header, without all the "representation metatada" (eg. the values of "Content-Type" and "Content-Encoding") may expose the communication to tampering.

"Digest" SHOULD always be used over a connection which provides integrity at transport layer that protects HTTP headers.

A "Digest" header using NOT RECOMMENDED digest-algorithms SHOULD NOT be used in signatures.

8.5. Message Truncation

...

8.6. Algorithm Agility

...

9. IANA Considerations

9.1. Establish the HTTP Digest Algorithm Values

This memo sets this spec to be the establishing document for the HTTP Digest Algorithm Values [4]

9.2. The "status" field in the HTTP Digest Algorithm Values

This memo adds the field "Status" to the HTTP Digest Algorithm Values [5] registry. The allowed values for the "Status" fields are described below.

Status Specify "standard", "experimental", "historic", "obsoleted", or "deprecated" according to the type and status of the primary document in which the algorithm is defined.

9.3. Obsolete "MD5" Digest Algorithm

This memo updates the "MD5" digest algorithm in the HTTP Digest Algorithm Values [6] registry:

- Digest Algorithm: MD5
- Description: As specified in Section 3.
- Status: As specified in Section 3.

9.4. Obsolete "SHA" Digest Algorithm

This memo updates the "SHA" digest algorithm in the HTTP Digest Algorithm Values [7] registry:

- Digest Algorithm: SHA
- Description: As specified in Section 3.
- Status: As specified in Section 3.

9.5. The "ID-SHA-256" Digest Algorithm

This memo registers the "ID-SHA-256" digest algorithm in the HTTP Digest Algorithm Values [8] registry:

- Digest Algorithm: ID-SHA-256
- Description: As specified in Section 3.
- Status: As specified in Section 3.

9.6. The "ID-SHA-512" Digest Algorithm

This memo registers the "ID-SHA-512" digest algorithm in the HTTP Digest Algorithm Values [9] registry:

- Digest Algorithm: ID-SHA-512
- Description: As specified in Section 3.
- Status: As specified in Section 3.

9.7. Changes compared to RFC5843

The status has been updated to "obsoleted" for both "SHA" and "MD5", and their descriptions states that those algorithms are NOT RECOMMENDED.

The status for all other algorithms have been updated to "standard".

The "ID-SHA-256" and "ID-SHA-512" algorithms have been added to the registry.

9.8. Want-Digest Header Field Registration

This section registers the "Want-Digest" header field in the "Permanent Message Header Field Names" registry ([RFC3864]).

Header field name: "Want-Digest"

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): Section 4.1 of this document

9.9. Digest Header Field Registration

This section registers the "Digest" header field in the "Permanent Message Header Field Names" registry ([RFC3864]).

Header field name: "Digest"

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): Section 4.2 of this document

10. References

10.1. Normative References

[CMU-836068]
Carnagie Mellon University, Software Engineering Institute, ., "MD5 Vulnerable to collision attacks", December 2008, <https://www.kb.cert.org/vuls/id/836068/>.

[FIPS180-1]
Department of Commerce, National., "NIST FIPS 180-1, Secure Hash Standard", April 1995, <http://csrc.nist.gov/fips/fip180-1.txt>.

[FIPS180-3]
Department of Commerce, National., "NIST FIPS 180-3, Secure Hash Standard", October 2008, <https://csrc.nist.gov/csrc/media/publications/fips/180/3/archive/2008-10-31/documents/fips180-3_final.pdf>.

[IACR-2019-459]
Inria, France; Nanyang Technological University, Singapore; Temasek Laboratories, Singapore, ., "From Collisions to Chosen-Prefix Collisions Application to Full SHA-1", May 2019, <https://eprint.iacr.org/2019/459.pdf>.

[NIST800-32]
Department of Commerce, National., "Introduction to Public Key Technology and the Federal PKI Infrastructure", February 2001, <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-32.pdf>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[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>.

[RFC3230] Mogul, J. and A. Van Hoff, "Instance Digests in HTTP", RFC 3230, DOI 10.17487/RFC3230, January 2002, <https://www.rfc-editor.org/info/rfc3230>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[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>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, DOI 10.17487/RFC5789, March 2010, <https://www.rfc-editor.org/info/rfc5789>.

[RFC5843] Bryan, A., "Additional Hash Algorithms for HTTP Instance Digests", RFC 5843, DOI 10.17487/RFC5843, April 2010, <https://www.rfc-editor.org/info/rfc5843>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7233] Fielding, R., Ed., Lafon, Y., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Range Requests", RFC 7233, DOI 10.17487/RFC7233, June 2014, <https://www.rfc-editor.org/info/rfc7233>.

[RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <https://www.rfc-editor.org/info/rfc7405>.

[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/info/rfc8174>.

[UNIX] The Open Group, ., "The Single UNIX Specification, Version 2 - 6 Vol Set for UNIX 98", February 1997.

10.2. Informative References

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC7396] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7396, DOI 10.17487/RFC7396, October 2014, <https://www.rfc-editor.org/info/rfc7396>.

[SRI] Akhawe, D., Braun, F., Marier, F., and J. Weinberger, "Subresource Integrity", June 2016.

10.3. URIs

[1] https://lists.w3.org/Archives/Public/ietf-http-wg/

[2] https://github.com/httpwg/http-extensions

[3] https://tools.ietf.org/html/rfc7231#section-3.1.2.1

[4] https://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml

[5] https://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml

[6] https://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml

[7] https://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml

[8] https://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml

[9] https://www.iana.org/assignments/http-dig-alg/http-dig-alg.xhtml

Appendix A. Change Log

RFC EDITOR PLEASE DELETE THIS SECTION.

Appendix B. FAQ

1. Why remove all references to content-md5?

   Those were unnecessary to understanding and using this spec.

2. Why remove references to instance manipulation?

   Those were unnecessary for correctly using and applying the spec. An example with Range Request is more than enough. This doc uses the term "partial representation" which should group all those cases.

3. How to use "Digest" with "PATCH" method?

   The PATCH verb brings some complexities (eg. about representation metadata headers, patch document format, ...),

   * PATCH entity-headers apply to the patch document and MUST NOT be applied to the target resource, see [RFC5789], Section 2.

   * servers shouldn't assume PATCH semantics for generic media types like "application/json" but should instead use a proper content-type, eg [RFC7396]

   * a "200 OK" response to a PATCH request would contain the digest of the patched item, and the etag of the new object. This behavior - tighly coupled to the application logic - gives the client low probability of guessing the actual outcome of this operation (eg. concurrent changes, ...)

4. Why remove references to delta-encoding?

   Unnecessary for a correct implementation of this spec. The revised spec can be nicely adapted to "delta encoding", but all the references here to delta encoding don't add anything to this RFC. Another job would be to refresh delta encoding.

5. Why remove references to Digest Authentication?

   This RFC seems to me completely unrelated to Digest Authentication but for the word "Digest".

6. What changes in "Want-Digest"?

We allow to use the "Want-Digest" in responses to advertise the supported digest-algorithms and the inability to accept requests with unsupported digest-algorithms.

7. Does this spec changes supported algorithms?

This RFC updates [RFC5843] which is still delegated for all algorithms updates, and adds two more algorithms: ID-SHA-256 and ID-SHA-512 which allows to send a checksum of a resource representation with no content codings applied.

Acknowledgements

The vast majority of this document is inherited from [RFC3230], so thanks to J. Mogul and A. Van Hoff for their great work. The original idea of refreshing this document arose from an interesting discussion with M. Nottingham, J. Yasskin and M. Thomson when reviewing the MICE content coding.

Authors' Addresses

Roberto Polli
Team Digitale, Italian Government

Email: robipolli@gmail.com

Lucas Pardue
Cloudflare

Email: lucaspardue.24.7@gmail.com