

HTTP  
Internet-Draft  
Intended status: Standards Track  
Expires: 23 January 2026

L. Pardue  
Cloudflare  
M. West  
Google  
22 July 2025

HTTP Unencoded Digest  
draft-ietf-httpbis-unencoded-digest-01

## Abstract

The Repr-Digest and Content-Digest integrity fields are subject to HTTP content coding considerations. There are some use cases that benefit from the unambiguous exchange of integrity digests of unencoded representation. The Unencoded-Digest and Want-Unencoded-Digest fields complement existing integrity fields for this purpose.

## About This Document

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

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-ietf-httpbis-unencoded-digest/>.

Discussion of this document takes place on the HTTP Working Group mailing list (<mailto:ietf-http-wg@w3.org>), which is archived at <https://lists.w3.org/Archives/Public/ietf-http-wg/>. Working Group information can be found at <https://httpwg.org/>.

Source for this draft and an issue tracker can be found at <https://github.com/httpwg/http-extensions/labels/unecoded-digest>.

## 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 23 January 2026.

## Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions and Definitions . . . . .	3
3. The Unencoded-Digest Field . . . . .	4
4. The Want-Unencoded-Digest Field . . . . .	5
5. Messages containing both Unencoded-Digest and Content-Encoding . . . . .	6
6. Integrity Fields are Complementary . . . . .	6
7. Security Considerations . . . . .	8
8. IANA Considerations . . . . .	8
9. Normative References . . . . .	9
Acknowledgments . . . . .	10
Authors' Addresses . . . . .	10

## 1. Introduction

The Repr-Digest and Content-Digest integrity fields defined in [DIGEST-FIELDS] are suitable for a range of use cases. However, because the fields are subject to HTTP content coding considerations, it is difficult to support use cases that could benefit from the exchange of integrity digests of the unencoded representation.

As a simple example, an application using HTTP might be presented with request or response representation data that has been transparently decoded. Attempting to verify the integrity of the data against the Repr-Digest would first require re-encoding that data using the same coding indicated by the Content-Encoding header field (Section 8.4 of [HTTP]), which is not always possible (see Section 6.5 of [DIGEST-FIELDS]).

Although receivers could feasibly re-encode data in order to carry out Repr-Digest validation, it might be impractical for certain kinds of environments. For instance, browsers tend to provide built-in support for transparent decoding but little support for encoding; while this could be done via the use of additional libraries it would create work in JavaScript that could contend with other activities. Even on the server side, the re-encoding of received data might not be acceptable; some coding algorithms are optimized towards efficient decoding at the cost of complex encoding. A Content-Encoding field value that indicates a series of encodings adds further complexity.

A more complex example involves HTTP Range Requests (Section 14 of [HTTP]), where a client fetches multiple partial representations from different origins and "stitches" them back into a whole. Unfortunately, if the origins apply different content coding, the Repr-Digest field will vary by the server's selected encoding (i.e. the Content-Encoding header field, Section 8.4 of [HTTP]). This provides a challenge for a client - in order to verify the integrity of the pieced-together whole it would need to remove the encoding of each part, combine them, and then encode the result in order to compare against one or more Repr-Digests.

The Accept-Encoding header field (Section 12.5.3 of [HTTP]) provides the means to indicate preferences for content coding. It is possible for an endpoint to indicate a preference for no encoding, for example by sending the "identity" token. However, codings often provide data compression that is advantageous. Disabling content coding in order to simplify integrity checking is possibly an unacceptable trade off.

For a variety of reasons, decoding and re-encoding content in order to benefit from HTTP integrity fields is not preferable. This specification defines the Unencoded-Digest and Want-Unencoded-Digest fields to support a simpler validation workflow in some scenarios where content coding is applied. These fields complement the other integrity fields defined in [DIGEST-FIELDS].

## 2. 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.

This document uses the Augmented BNF defined in [RFC5234] and updated by [RFC7405]. This includes the rules: LF (line feed)

This document uses the following terminology from Section 3 of [STRUCTURED-FIELDS] to specify syntax and parsing: Byte Sequence, Dictionary, and Integer.

The definitions "representation", "selected representation", "representation data", "representation metadata", and "content" in this document are to be interpreted as described in [HTTP].

"Integrity fields" is the collective term for Content-Digest, Repr-Digest, and Unencoded-Digest.

"Integrity preference fields" is the collective term for Want-Repr-Digest, Want-Content-Digest, and Want-Unencoded-Digest.

### 3. The Unencoded-Digest Field

The Unencoded-Digest HTTP field can be used in requests and responses to communicate digests that are calculated using a hashing algorithm applied to the representation with no content coding (Section 8.4.1 of [HTTP]).

Apart from the content coding concerns, Unencoded-Digest behaves similarly to Repr-Digest (Section 3 of [DIGEST-FIELDS]). In the absence of content coding, Unencoded-Digest is identical to Repr-Digest.

Unencoded-Digest is a Dictionary (see Section 3.2 of [STRUCTURED-FIELDS]) where each:

- \* key conveys the hashing algorithm (see Section 5 of [DIGEST-FIELDS]) used to compute the digest;
- \* value is a Byte Sequence (Section 3.3.5 of [STRUCTURED-FIELDS]), that conveys an encoded version of the byte output produced by the digest calculation.

Each Dictionary value can have zero or more Parameters (Section 3.1.2 of [STRUCTURED-FIELDS]). This specification does not define any Parameters but extensions MAY define them. Unknown Parameters MUST be ignored.

For example:

NOTE: '\ ' line wrapping per RFC 8792

```
Unencoded-Digest: \
  sha-512=:YMAam51Jz/jOATT6/zvHrLVgOYTGFyld6GJiOHTohq4yP+pgk4vf2aCs\
  yRZotw8Mjkm7iw7yZ/WkppmM44T3qg==:
```

The Dictionary type can be used, for example, to attach multiple digests calculated using different hashing algorithms in order to support a population of endpoints with different or evolving capabilities. Such an approach could support transitions away from weaker algorithms (see Section 6.6 of [DIGEST-FIELDS]).

NOTE: '\ ' line wrapping per RFC 8792

```
Unencoded-Digest: \
  sha-256=:d435Qo+nKZ+gLcUhn7GQtQ72hiBVAgqoLsZnZPiTGPK=:,\
  sha-512=:YMAam5lJz/jOATT6/zvHrLVgOYTGFyld6GJiOHTohq4yP+pgk4vf2aCs\
  yRZotw8Mjkm7iw7yZ/WkppmM44T3qg==:
```

A recipient MAY ignore any or all digests. Application-specific behavior or local policy MAY set additional constraints on the processing and validation practices of the conveyed digests. The security considerations cover some of the issues related to ignoring digests (see Section 6.6 of [DIGEST-FIELDS]) and validating multiple digests (see Section 6.7 of [DIGEST-FIELDS]).

A sender MAY send a digest without knowing whether the recipient supports a given hashing algorithm. A sender MAY send a digest if it knows the recipient will ignore it.

Unencoded-Digest can be sent in a trailer section. In this case, Unencoded-Digest MAY be merged into the header section; see Section 6.5.1 of [HTTP].

#### 4. The Want-Unencoded-Digest Field

Want-Unencoded-Digest is an integrity preference field; see Section 4 of [DIGEST-FIELDS]. It indicates that the sender would like to receive (via the Unencoded-Digest field) a representation digest on messages associated with the request URI and representation metadata where no content coding is applied.

If Want-Unencoded-Digest is used in a response, it indicates that the server would like the client to provide the Unencoded-Digest field on future requests.

Want-Unencoded-Digest is only a hint. The receiver of the field can ignore it and send an Unencoded-Digest field using any algorithm or omit one entirely. It is not a protocol error if preferences are ignored. Applications that use Unencoded-Digest and Want-Unencoded-Digest can define expectations or constraints that operate in addition to this specification.

Want-Unencoded-Digest is of type Dictionary where each:

- \* key conveys the hashing algorithm;
- \* value is an Integer (Section 3.3.1 of [STRUCTURED-FIELDS]) that conveys an ascending, relative, weighted preference. It must be in the range 0 to 10 inclusive. 1 is the least preferred, 10 is the most preferred, and a value of 0 means "not acceptable".

Each Dictionary value can have zero or more Parameters (Section 3.1.2 of [STRUCTURED-FIELDS]). This specification does not define any Parameters but extensions MAY define them. Unknown Parameters MUST be ignored.

Examples:

Want-Unencoded-Digest: sha-256=1

Want-Unencoded-Digest: sha-512=3, sha-256=10, unixsum=0

## 5. Messages containing both Unencoded-Digest and Content-Encoding

Digests delivered through Unencoded-Digest apply to the unencoded representation. If a message is received with content coding, a recipient needs to decode the message in order to calculate the digest that can subsequently be used for validation. If multiple content codings are applied, the recipient needs to decode all encodings in order before validation.

## 6. Integrity Fields are Complementary

Integrity fields can be used in combination to address different and complementary needs, particularly the cases described in Section 1.

In the following examples, the unencoded response data is the string "An unexceptional string" following by an LF.

The first example demonstrates a request that uses content negotiation.

```
GET /boringstring HTTP/1.1
Host: example.org
Accept-Encoding: gzip
```

Figure 1: GET request with content negotiation

The server responds with the full GZIP-encoded representation. The Repr-Digest and Unencoded-Digest therefore differ.

NOTE: '\' line wrapping per RFC 8792

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Repr-Digest: \
  sha-256=:XyjuEuFblP5rqc2le3vQm7M96DwZhvmOwqHLu2xVpY4=:
Unencoded-Digest: \
  sha-256=:5Bv3NIx05BPnh0jMph6v1RJ5Q7kl9LKmtQxmvc9+Z7Y=:
```

```
1f 8b 08 00 79 1f 08 64 00 ff
73 cc 53 28 cd 4b ad 48 4e 2d
28 c9 cc cf 4b cc 51 28 2e 29
ca cc 4b e7 02 00 7e af 07 44
18 00 00 00
```

Figure 2: GET response with GZIP-encoded content

The second example demonstrates a range request with content negotiation.

```
GET /boringstring HTTP/1.1
Host: example.org
Accept-Encoding: gzip
Range: bytes=0-10
```

Figure 3: Range request with content negotiation

The server responds with a 206 Partial Content response using GZIP encoding, it has three different Integrity fields. The Content-Digest relates to the response message content that can be used to validate the integrity of the received part. Repr-Digest and Unencoded-Digest can be used later once the entire object is reconstructed. The choice of which to use is left to the application that would consider a range of factors outside the scope of this document.

NOTE: '\ ' line wrapping per RFC 8792

```
HTTP/1.1 206 Partial Content
Content-Encoding: gzip
Content-Range: bytes 0-9/44
Content-Digest: \
  sha-256=:SotB7Pa5A7iHSBdh9mg1Ev/ktAzrxU4Z8ldcCIUyfiI4=:
Repr-Digest: \
  sha-256=:XyjvEuFb1P5rqc2le3vQm7M96DwZhvmOwqHLu2xVpY4=:
Unencoded-Digest: \
  sha-256=:5Bv3NIx05BPnh0jMph6v1RJ5Q7kl9LKmtQxmvc9+Z7Y=:

1f 8b 08 00 79 1f 08 64 00 ff
```

Figure 4: Partial response with GZIP encoding

## 7. Security Considerations

All the same considerations documented in [DIGEST-FIELDS] apply.

This document introduces a further consideration related to the process of validation when an HTTP message contains both Content-Encoding and Unencoded-Digest (Section 5). In order to validate the Unencoded-Digest, encoded content needs to be decoded. This provides an opportunity for an attacker to direct malicious data into a decoder. One possible mitigation would be to also provide a Content-Digest or Repr-Digest in the message, allowing for validation of the received bytes before further processing. An attacker that can substitute various parts of an HTTP message presents several risks, Sections 6.1, 6.2 and 6.3 of [DIGEST-FIELDS] describe relevant considerations and mitigations.

## 8. IANA Considerations

Should this document be adopted and achieve working group consensus, IANA is asked to update the "Hypertext Transfer Protocol (HTTP) Field Name Registry" [HTTP] as shown in the table below:



Field Name	Status	Structured Type	Reference
Unencoded-Digest	permanent	Dictionary	Section 3 of this document
Want-Unencoded-Digest	permanent	Dictionary	Section 4 of this document

Table 1: Hypertext Transfer Protocol (HTTP) Field Name Registry Update

## 9. Normative References

### [DIGEST-FIELDS]

Polli, R. and L. Pardue, "Digest Fields", RFC 9530, DOI 10.17487/RFC9530, February 2024, <<https://www.rfc-editor.org/rfc/rfc9530>>.

### [HTTP]

Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <<https://www.rfc-editor.org/rfc/rfc9110>>.

### [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>>.

### [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/rfc/rfc5234>>.

### [RFC7405]

Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <<https://www.rfc-editor.org/rfc/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/rfc/rfc8174>>.

### [STRUCTURED-FIELDS]

Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 9651, DOI 10.17487/RFC9651, September 2024, <<https://www.rfc-editor.org/rfc/rfc9651>>.

## Acknowledgments

Early drafts of [DIGEST-FIELDS] included a mechanism to support the exchange of digests where no content coding is applied, which was removed before publication. While the design here is different, it is motivated by discussion of the previous design in the HTTP WG. The motivating use cases still mostly apply identically.

## Authors' Addresses

Lucas Pardue  
Cloudflare  
Email: [lucas@lucaspardue.com](mailto:lucas@lucaspardue.com)

Mike West  
Google  
Email: [mkwst@google.com](mailto:mkwst@google.com)