Content-Digest header

- Syntax
- Directives
- Examples
  - A SHA-256 Content-Digest in a response
  - Identical Content-Digest and Repr-Digest values
  - Diverging Content-Digest and Repr-Digest values
  - Digest of a gzip-encoded representation
  - Content-Digest handling of no content
  - User-agent sending digests in requests
- Specifications
- Browser compatibility
- See also

The HTTP `Content-Digest` request and response header provides a digest calculated using a hashing algorithm applied to the message content. A recipient can use the `Content-Digest` to validate the HTTP message content for integrity purposes.

The Want-Content-Digest field lets a sender request a `Content-Digest` along with their hashing algorithm preferences. A content digest will differ based on Content-Encoding and Content-Range, but not Transfer-Encoding.

In certain cases, a Repr-Digest can be used to validate the integrity of partial or multipart messages against the full representation. For example, in range requests, a `Repr-Digest` will always have the same value if only the requested byte ranges differ, whereas the content digest will be different for each part. For this reason, a `Content-Digest` is identical to a Repr-Digest when a representation is sent in a single message.

Header type	[Request header](https://developer.mozilla.org/en-US/docs/Glossary/Request%20header), [Response header](https://developer.mozilla.org/en-US/docs/Glossary/Response%20header), [Representation header](https://developer.mozilla.org/en-US/docs/Glossary/Representation%20header)
[Forbidden request header](https://developer.mozilla.org/en-US/docs/Glossary/Forbidden%20request%20header)	No

Syntax

```
Content-Digest: <digest-algorithm>=<digest-value>


// Multiple digest algorithms
Content-Digest: <digest-algorithm>=<digest-value>,<digest-algorithm>=<digest-value>, …
```

`Content-Digest` is a *structured field dictionary* (Structured Field Values for HTTP), whose keys are `<digest-algorithm>` and values are `<digest-value>`.

Directives

- `<digest-algorithm>`
  - : The algorithm used to create a digest of the message content. Only two registered digest algorithms are considered secure: `sha-512` and `sha-256`. The insecure (legacy) registered digest algorithms are: `md5`, `sha` (SHA-1), `unixsum`, `unixcksum`, `adler` (ADLER32) and `crc32c`.
- `<digest-value>`
  - : The digest of the message content using the `<digest-algorithm>`, base64-encoded and wrapped in colons (`:`, ASCII 0x3A). This encoding is referred to as a Byte Sequence in the specification.

Examples

In all of the examples, endpoints are configured to send unsolicited digest headers. The Want-Content-Digest and Want-Repr-Digest fields could optionally be used by a sender to request a `Content-Digest`

or `Repr-Digest` along with their hashing algorithm preferences.

A SHA-256 Content-Digest in a response

A user-agent requests a resource:

```
GET /items/123 HTTP/1.1
Host: example.com
```

The server responds with a `Content-Digest` of the message content using the SHA-256 algorithm. The digest is calculated over the exact bytes of the message body, `{"hello": "mdn"}` (16 bytes, explicitly not including any trailing line break):

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 16
Content-Digest: sha-256=:bMGjiT1wkArOzyB9ReAdpW51FV4mHlQygPXGp+TtzG4=:

{"hello": "mdn"}
```

Identical Content-Digest and Repr-Digest values

A user-agent requests a resource:

```
GET /items/123 HTTP/1.1
Host: example.com
```

The server responds with a `Content-Digest` and `Repr-Digest` of the message content using the SHA-256 algorithm. The `Repr-Digest` and `Content-Digest` fields have matching values because they are calculated using the same algorithm over the same bytes, `{"hello": "mdn"}` (16 bytes), and in this case the entire representation is sent in one message:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 16
Content-Digest: sha-256=:bMGjiT1wkArOzyB9ReAdpW51FV4mHlQygPXGp+TtzG4=:
Repr-Digest: sha-256=:bMGjiT1wkArOzyB9ReAdpW51FV4mHlQygPXGp+TtzG4=:
```

```
{"hello": "mdn"}
```

Diverging Content-Digest and Repr-Digest values

A user-agent requests only part of a resource using a range request:

```
GET /items/123 HTTP/1.1
Host: example.com
Range: bytes=0-7
```

The server returns a 206 Partial Content response containing only the requested bytes, `{"hello"` (8 bytes), as the message content. `Content-Digest` covers only those bytes, while `Repr-Digest` still covers the entire representation, `{"hello": "mdn"}` (16 bytes), so the two values differ:

```
HTTP/1.1 206 Partial Content
Content-Type: application/json
Content-Range: bytes 0-7/16
Content-Digest: sha-256=:pKQv0IAKChzGfyfxu5TNqcnvxIzaG4XICf6NQnB1YhY=:
Repr-Digest: sha-256=:bMGjiT1wkArOzyB9ReAdpW51FV4mHlQygPXGp+TtzG4=:
```

Digest of a gzip-encoded representation

In this request the client uses the Accept-Encoding header to indicate that it accepts gzip compression:

```
GET /items/123 HTTP/1.1
Host: example.com
Accept-Encoding: gzip
```

The server response includes the Content-Encoding header, indicating that the message bytes are from the gzip representation of the resource. The digest is calculated over the gzip-encoded bytes instead of the original unencoded text. Here, the 16-byte JSON body `{"hello": "mdn"}` is gzip-compressed to a 36-byte representation, and `Content-Digest` and `Repr-Digest` are calculated over those 36 bytes (shown here as hex for readability):

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Encoding: gzip
Content-Length: 36
Content-Digest: sha-256=:6Gx6u1ZhhahDLs06Zc6ZEqXxUy8RNjy18CaMucjKOFk=:
Repr-Digest: sha-256=:6Gx6u1ZhhahDLs06Zc6ZEqXxUy8RNjy18CaMucjKOFk=:
1F 8B 08 00 00 00 00 00 02 FF AB 56 CA 48 CD C9 C9 57 B2 52 50 CA 4D C9 53 AA 05 00 35 D8 1D
91 10 00 00 00
```

Content-Digest handling of no content

If the same resource is requested with a HEAD method instead of a GET, the response has no content:

```
HEAD /items/123 HTTP/1.1
Host: example.com
```

The `Repr-Digest` value is the same as before, since it always applies to the full representation, `{"hello": "mdn"}`. However, the server will not send any content in the response and can omit the `Content-Digest` header:

```
HTTP/1.1 200 OK
Content-Type: application/json
Repr-Digest: sha-256=:bMGjiT1wkArOzyB9ReAdpW51FV4mHlQygPXGp+TtzG4=:
```

Instead of omitting `Content-Digest` when there is no content, a server can explicitly compute it over an empty string. Per Section 6.3 of RFC 9530, this lets a recipient, particularly when the digest is covered by an HTTP message signature, verify that no content was added or removed, rather than only that the header was left out:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Digest: sha-256=:47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=:
Repr-Digest: sha-256=:bMGjiT1wkArOzyB9ReAdpW51FV4mHlQygPXGp+TtzG4=:
```

User-agent sending digests in requests

In the following example, a user-agent sends a digest of the message content using SHA-512. The digest is calculated over the exact bytes of the message body,

`{"recipient":"Alex","amount":900000000}` (39 bytes, explicitly not including any trailing line break). Since the entire representation is sent in this single request, `Content-Digest` and `Repr-Digest` have the same value:

```
POST /bank_transfer HTTP/1.1
Host: example.com
Content-Type: application/json
Content-Length: 39
Content-Digest: sha-
512=:PlrIZYU3M76B30wGsL0h6O79BoxHTdAG+RnMPjOyECTSJCN/KnYdOrSCCWjxV3ckkyvdRmZ52//M3WbehCXcPw==:
Repr-Digest: sha-
512=:PlrIZYU3M76B30wGsL0h6O79BoxHTdAG+RnMPjOyECTSJCN/KnYdOrSCCWjxV3ckkyvdRmZ52//M3WbehCXcPw==:

{"recipient":"Alex","amount":900000000}
```

Specifications

Browser compatibility

This header has no specification-defined browser integration ("browser compatibility" does not apply). Developers can set and get HTTP headers using `fetch()` in order to provide application-specific implementation behavior.

See also

- Want-Content-Digest header to request a content digest
- Repr-Digest, Want-Repr-Digest representation digest headers
- ETag
- Digital Signatures for APIs SDK guide uses `Content-Digest`s for digital signatures in HTTP calls (developer.ebay.com)

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:35:38 UTC