Transfer-Encoding

The **`Transfer-Encoding`** header specifies the form of encoding used to safely transfer the entity to the user.

> HTTP/2 doesn't support HTTP 1.1's chunked transfer encoding mechanism, as it provides its own, more efficient, mechanisms for data streaming.

`Transfer-Encoding` is a hop-by-hop header, that is applied to a message between two nodes, not to a resource itself. Each segment of a multi-node connection can use different `Transfer-Encoding` values. If you want to compress data over the whole connection, use the end-to-end `Content-Encoding` header instead.

When present on a response to a `HEAD` request that has no body, it indicates the value that would have applied to the corresponding `GET` message.

Header type	Response header
Forbidden header name	yes

Syntax

```
Transfer-Encoding: chunked
Transfer-Encoding: compress
Transfer-Encoding: deflate
Transfer-Encoding: gzip
Transfer-Encoding: identity

// Several values can be listed, separated by a comma
Transfer-Encoding: gzip, chunked
```

Directives

chunked

Data is sent in a series of chunks. The `Content-Length` header is omitted in this case and at the beginning of each chunk you need to add the length of the current chunk in hexadecimal format, followed by `'\r\n'` and then the chunk itself, followed by another

'\r\n'. The terminating chunk is a regular chunk, with the exception that its length is zero. It is followed by the trailer, which consists of a (possibly empty) sequence of entity header fields.

compress

A format using the Lempel-Ziv-Welch (LZW) algorithm. The value name was taken from the UNIX *compress* program, which implemented this algorithm.
Like the compress program, which has disappeared from most UNIX distributions, this content-encoding is used by almost no browsers today, partly because of a patent issue (which expired in 2003).

deflate

Using the zlib structure (defined in RFC 1950), with the *deflate* compression algorithm (defined in RFC 1951).

gzip

A format using the Lempel-Ziv coding (LZ77), with a 32-bit CRC. This is originally the format of the UNIX *gzip* program. The HTTP/1.1 standard also recommends that the servers supporting this content-encoding should recognize `x-gzip` as an alias, for compatibility purposes.

identity

Indicates the identity function (i.e. no compression, nor modification). This token, except if explicitly specified, is always deemed acceptable.

Examples

Chunked encoding

Chunked encoding is useful when larger amounts of data are sent to the client and the total size of the response may not be known until the request has been fully processed. For example, when generating a large HTML table resulting from a database query or when transmitting large images. A chunked response looks like this:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Transfer-Encoding: chunked

7\r\n
Mozilla\r\n
9\r\n
Developer\r\n
7\r\n
Network\r\n
0\r\n
\r\n
```

Specifications

Specification	Title

RFC 7230, section 3.3.1: Transfer-Encoding	Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing

Browser compatibility

Take this quick survey to help us improve our browser compatibility tables

Basic support	
Chrome	Yes
Edge	Yes
Firefox	Yes
IE	Yes
Opera	Yes
Safari	Yes
WebView Android	Yes
Chrome Android	Yes
Edge Mobile	Yes
Firefox Android	Yes
Opera Android	Yes
Safari iOS	Yes

Samsung Internet Android	Yes

Full support

See also

- `Accept-Encoding`
- `Content-Encoding`
- `Content-Length`
- Header fields that regulate the use of trailers: `TE` (requests) and `Trailer` (responses).
- Chunked transfer encoding