Abstract

This specification defines a mechanism for using designated HTTP responses as an external dictionary for future HTTP responses for compression schemes that support using external dictionaries (e.g., Brotli (RFC 7932) and Zstandard (RFC 8878)).

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-compression-dictionary/.

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/compression-dictionary.

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 November 2024.
Table of Contents

1. Introduction
   1.1. Notational Conventions

2. Dictionary Negotiation
   2.1. Use-As-Dictionary
       2.1.1. match
       2.1.2. match-dest
       2.1.3. id
       2.1.4. type
       2.1.5. Examples
   2.2. Available-Dictionary
       2.2.1. Dictionary freshness requirement
       2.2.2. Dictionary URL matching
       2.2.3. Multiple matching dictionaries
   2.3. Dictionary-ID

3. The 'compression-dictionary' Link Relation Type

4. Dictionary-Compressed Brotli

5. Dictionary-Compressed Zstandard

6. Negotiating the content encoding
   6.1. Accept-Encoding
   6.2. Content-Encoding

7. IANA Considerations
   7.1. Content Encoding
   7.2. Header Field Registration
   7.3. Link Relation Registration

8. Compatibility Considerations

9. Security Considerations
   9.1. Changing content
   9.2. Reading content
   9.3. Security Mitigations
       9.3.1. Cross-origin protection
       9.3.2. Response readability

10. Privacy Considerations
11. References

11.1. Normative References

11.2. Informative References

Authors' Addresses

1. Introduction

This specification defines a mechanism for using designated [HTTP] responses as an external dictionary for future HTTP responses for compression schemes that support using external dictionaries (e.g., Brotili [RFC7932] and Zstandard [RFC8878]).

This document describes the HTTP headers used for negotiating dictionary usage and registers media types for content encoding Brotili and Zstandard using a negotiated dictionary.

1.1. 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] when, and only when, they appear in all capitals, as shown here.

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

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

This document uses the line folding strategies described in [FOLDING].

This document also uses terminology from [HTTP] and [HTTP-CACHING].

2. Dictionary Negotiation

2.1. Use-As-Dictionary

When responding to a HTTP Request, a server can advertise that the response can be used as a dictionary for future requests for URLs that match the rules specified in the Use-As-Dictionary response header.

The Use-As-Dictionary response header is a Structured Field [STRUCTURED-FIELDS] Dictionary with values for "match", "match-dest", "id", and "type".
2.1.1. match

The "match" value of the Use-As-Dictionary header is a String value that provides the URL Pattern [URLPattern] to use for request matching.

The URL Pattern used for matching does not support using Regular expressions.

The following algorithm will return TRUE for a valid match pattern and FALSE for an invalid pattern that MUST NOT be used:

1. Let MATCH be the value of "match" for the given dictionary.
2. Let URL be the URL of the dictionary request.
3. Let PATTERN be a URL Pattern [URLPattern] constructed by setting input=MATCH, and baseURL=URL.
4. If PATTERN has regexp groups then return FALSE.
5. Return True.

The "match" value is required and MUST be included in the Use-As-Dictionary Dictionary for the dictionary to be considered valid.

2.1.2. match-dest

The "match-dest" value of the Use-As-Dictionary header is an Inner List of String values that provides a list of request destinations for the dictionary to match (https://fetch.spec.whatwg.org/#concept-request-destination).

An empty list for "match-dest" MUST match all destinations.

For clients that do not support request destinations, the client MUST treat it as an empty list and match all destinations.

The "match-dest" value is optional and defaults to an empty list.

2.1.3. id

The "id" value of the Use-As-Dictionary header is a String value that specifies a server identifier for the dictionary. If an "id" value is present and has a string length longer than zero then it MUST be sent to the server in a "Dictionary-ID" request header when the dictionary is advertised as being available.

The server identifier MUST be treated as an opaque string by the client.
The server identifier **MUST NOT** be relied upon by the server to guarantee the contents of the dictionary. The dictionary hash **MUST** be validated before use.

The "id" value string length (after any decoding) supports up to 1024 characters.

The "id" value is optional and defaults to the empty string.

### 2.1.4. type

The "type" value of the Use-As-Dictionary header is a Token value that describes the file format of the supplied dictionary.

"raw" is the only defined dictionary format which represents an unformatted blob of bytes suitable for any compression scheme to use.

If a client receives a dictionary with a type that it does not understand, it **MUST NOT** use the dictionary.

The "type" value is optional and defaults to raw.

### 2.1.5. Examples

#### 2.1.5.1. Path Prefix

A response that contained a response header:

```
NOTE: '\ line wrapping per RFC 8792

Use-As-Dictionary: \n  match="/product/**", match-dest=("document")
```

Would specify matching any document request for a URL with a path prefix of /product/ on the same [Origin] as the original request.

#### 2.1.5.2. Versioned Directories

A response that contained a response header:

```
Use-As-Dictionary: match="/app/*/main.js"
```

Would match main.js in any directory under /app/ and expiring as a dictionary in one year.

### 2.2. Available-Dictionary

When a HTTP client makes a request for a resource for which it has an appropriate dictionary, it can add a "Available-Dictionary"
request header to the request to indicate to the server that it has a dictionary available to use for compression.

The "Available-Dictionary" request header is a Structured Field [STRUCTURED-FIELDS] Byte Sequence containing the [SHA-256] hash of the contents of a single available dictionary.

The client **MUST** only send a single "Available-Dictionary" request header with a single hash value for the best available match that it has available.

For example:

```
Available-Dictionary: :pZGm1Av0IEBKARczz7exkNYsZb8LzaMrV7J32a2fFG4=:
```

### 2.2.1. Dictionary freshness requirement

To be considered as a match, the dictionary resource **MUST** be either fresh [HTTP-CACHING] or allowed to be served stale (see eg [RFC5861]).

### 2.2.2. Dictionary URL matching

When a dictionary is stored as a result of a "Use-As-Dictionary" directive, it includes "match" and "match-dest" strings that are used to match an outgoing request from a client to the available dictionaries.

Dictionaries **MUST** have been served from the same {Origin} as the outgoing request to match.

To see if an outbound request matches a given dictionary, the following algorithm will return TRUE for a successful match and FALSE for no-match:

1. If the current client supports request destinations:
   
   *Let DEST be the value of "match-dest" for the given dictionary.

   *Let REQUEST_DEST be the value of the destination for the current request.

   *If DEST is not an empty list and if REQUEST_DEST is not in the DEST list of destinations, return FALSE

2. Let BASEURL be the URL of the dictionary request.

3. Let URL represent the URL of the outbound request being checked.
4. If the \{Origin\} of BASEURL and the \{Origin\} of URL are not the same, return FALSE.

5. Let MATCH be the value of "match" for the given dictionary.

6. Let PATTERN be a URL Pattern [URLPattern] constructed by setting input=MATCH, and baseURL=BASEURL.

7. Return the result of running the "test" method of PATTERN with input=URL.

2.2.3. Multiple matching dictionaries

When there are multiple dictionaries that match a given request URL, the client **MUST** pick a single dictionary using the following rules:

1. For clients that support request destinations, a dictionary that specifies and matches a "match-dest" takes precedence over a match that does not use a destination.

2. Given equivalent destination precedence, the dictionary with the longest "match" takes precedence.

3. Given equivalent destination and match length precedence, the most recently fetched dictionary takes precedence.

2.3. Dictionary-ID

When a HTTP client makes a request for a resource for which it has an appropriate dictionary and the dictionary was stored with a server-provided "id" in the Use-As-Dictionary response then the client **MUST** echo the stored "id" in a "Dictionary-ID" request header.

The "Dictionary-ID" request header is a Structured Field [STRUCTURED-FIELDS] String of up to 1024 characters (after any decoding) and **MUST** be identical to the server-provided "id".

For example:

Available-Dictionary: :pZGm1Av0IEBKARczz7exkNYsZb8LzaMrV7J32a2fFG4=: Dictionary-ID: "/v1/main.js 33a64df551425fccc55e4d42a148795d9f25f89d4"

3. The 'compression-dictionary' Link Relation Type

This specification defines the 'compression-dictionary' link relation type [WEB-LINKING] that provides a mechanism for a HTTP response to provide a URL for a compression dictionary that is related to, but not directly used by the current HTTP response.
The 'compression-dictionary' link relation type indicates that fetching and caching the specified resource is likely to be beneficial for future requests. The response to some of those future requests are likely to be able to use the indicated resource as a compression dictionary.

Clients can fetch the provided resource at a time that they determine would be appropriate.

The response to the fetch for the compression dictionary needs to include a "Use-As-Dictionary" and caching response headers for it to be usable as a compression dictionary. The link relation only provides the mechanism for triggering the fetch of the dictionary.

The following example shows a link to a resource at https://example.org/dict.dat that is expected to produce a compression dictionary:

```
Link: <https://example.org/dict.dat>; rel="compression-dictionary"
```

4. Dictionary-Compressed Brotli

The "dcb" content encoding identifies a resource that is a "Dictionary-Compressed Brotli" stream.

A "Dictionary-Compressed Brotli" stream has a fixed header that is followed by a Shared Brotli [SHARED-BROTLI] stream. The header consists of a fixed 4 byte sequence and a 32 byte hash of the external dictionary that was used. The Shared Brotli stream is created using the referenced external dictionary and a compression window that is at most 16 MB in size.

The 36-byte fixed header is as follows:

**Magic_Number**: 4 fixed bytes: 0xff, 0x44, 0x43, 0x42.

**SHA_256_Hash**: 32 Bytes. SHA-256 hash digest of the dictionary [SHA-256].

Clients that announce support for dcb content encoding **MUST** be able to decompress resources that were compressed with a window size of up to 16 MB.

With Brotli compression, the full dictionary is available during compression and decompression independent of the compression window, allowing for delta-compression of resources larger than the compression window.
5. Dictionary-Compressed Zstandard

The "dcz" content encoding identifies a resource that is a "Dictionary-Compressed Zstandard" stream.

A "Dictionary-Compressed Zstandard" stream is a binary stream that starts with a 40-byte fixed header and is followed by a Zstandard [RFC8878] stream of the response that has been compressed with an external dictionary.

The 40-byte header consists of a fixed 8-byte sequence followed by the 32-byte SHA-256 hash of the external dictionary that was used to compress the resource:

**Magic_Number:** 8 fixed bytes: 0x5e, 0x2a, 0x4d, 0x18, 0x20, 0x00, 0x00, 0x00.

**SHA_256_Hash:** 32 Bytes. SHA-256 hash digest of the dictionary [SHA-256].

The 40-byte header is a Zstandard skippable frame (little-endian 0x184D2A5E) with a 32-byte length (little-endian 0x00000020) that is compatible with existing Zstandard decoders.

Clients that announce support for dcz content encoding **MUST** be able to decompress resources that were compressed with a window size of at least 8 MB or 1.25 times the size of the dictionary, which ever is greater, up to a maximum of 128 MB.

The window size used will be encoded in the content (currently, this can be expressed in powers of two only) and it **MUST** be lower than this limit. An implementation **MAY** treat a window size that exceeds the limit as a decoding error.

With Zstandard compression, the full dictionary is available during compression and decompression until the size of the input exceeds the compression window. Beyond that point the dictionary becomes unavailable. Using a compression window that is 1.25 times the size of the dictionary allows for full delta compression of resources that have grown by 25% between releases while still giving the client control over the memory it will need to allocate for a given response.

6. Negotiating the content encoding

When a compression dictionary is available for use for a given request, the encoding to be used is negotiated through the regular mechanism for negotiating content encoding in HTTP through the "Accept-Encoding" request header and "Content-Encoding" response header.
The dictionary to use is negotiated separately and advertised in the "Available-Dictionary" request header.

6.1. Accept-Encoding

The client adds the content encodings that it supports to the "Accept-Encoding" request header. e.g.:

Accept-Encoding: gzip, deflate, br, zstd, dcb, dcz

6.2. Content-Encoding

If a server supports one of the dictionary encodings advertised by the client and chooses to compress the content of the response using the dictionary that the client has advertised then it sets the "Content-Encoding" response header to the appropriate value for the algorithm selected. e.g.:

Content-Encoding: dcb

If the response is cacheable, it **MUST** include a "Vary" header to prevent caches serving dictionary-compressed resources to clients that don't support them or serving the response compressed with the wrong dictionary:

Vary: accept-encoding, available-dictionary

7. IANA Considerations

7.1. Content Encoding

IANA is asked to enter the following into the "HTTP Content Coding Registry" registry ([HTTP]):

*Name: dcb

*Description: "Dictionary-Compressed Brotli" data format.

*Reference: This document

*Notes: Section 4

IANA is asked to enter the following into the "HTTP Content Coding Registry" registry ([HTTP]):

*Name: dcz

*Description: "Dictionary-Compressed Zstandard" data format.

*Reference: This document
7.2. Header Field Registration

IANA is asked to update the "Hypertext Transfer Protocol (HTTP) Field Name Registry" registry ([HTTP]) according to the table below:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use-As-Dictionary</td>
<td>permanent</td>
<td>Section 2.1 of this document</td>
</tr>
<tr>
<td>Available-Dictionary</td>
<td>permanent</td>
<td>Section 2.2 of this document</td>
</tr>
<tr>
<td>Dictionary-ID</td>
<td>permanent</td>
<td>Section 2.3 of this document</td>
</tr>
</tbody>
</table>

Table 1

7.3. Link Relation Registration

IANA is asked to update the "Link Relation Type Registry" registry ([WEB-LINKING]):

*Relation Name: compression-dictionary

*Description: Refers to a compression dictionary used for content encoding.

*Reference: This document, Section 3

8. Compatibility Considerations

To minimize the risk of middle-boxes incorrectly processing dictionary-compressed responses, compression dictionary transport MUST only be used in secure contexts (HTTPS).

9. Security Considerations

The security considerations for Brotli [RFC7932], Shared Brotli [SHARED-BROTLI] and Zstandard [RFC8878] apply to the dictionary-based versions of the respective algorithms.

9.1. Changing content

The dictionary must be treated with the same security precautions as the content, because a change to the dictionary can result in a change to the decompressed content.

The dictionary is validated using a SHA-256 hash of the content to make sure that the client and server are both using the same dictionary. The strength of the SHA-256 hash algorithm isn't explicitly needed to counter attacks since the dictionary is being served from the same origin as the content. That said, if a weakness is discovered in SHA-256 and it is determined that the dictionary negotiation should use a different hash algorithm, the "Use-As-
Dictionary" response header can be extended to specify a different algorithm and the server would just ignore any "Available-Dictionary" requests that do not use the updated hash.

9.2. Reading content

The CRIME attack shows that it's a bad idea to compress data from mixed (e.g. public and private) sources -- the data sources include not only the compressed data but also the dictionaries. For example, if you compress secret cookies using a public-data-only dictionary, you still leak information about the cookies.

Not only can the dictionary reveal information about the compressed data, but vice versa, data compressed with the dictionary can reveal the contents of the dictionary when an adversary can control parts of data to compress and see the compressed size. On the other hand, if the adversary can control the dictionary, the adversary can learn information about the compressed data.

9.3. Security Mitigations

If any of the mitigations do not pass, the client MUST drop the response and return an error.

9.3.1. Cross-origin protection

To make sure that a dictionary can only impact content from the same origin where the dictionary was served, the URL Pattern used for matching a dictionary to requests (Section 2.1.1) is guaranteed to be for the same origin that the dictionary is served from.

9.3.2. Response readability

For clients, like web browsers, that provide additional protection against the readability of the payload of a response and against user tracking, additional protections MUST be taken to make sure that the use of dictionary-based compression does not reveal information that would not otherwise be available.

In these cases, dictionary compression MUST only be used when both the dictionary and the compressed response are fully readable by the client.

In browser terms, that means that both are either same-origin to the context they are being fetched from or that the response is cross-origin and passes the CORS check (https://fetch.spec.whatwg.org/#cors-check).
### 9.3.2.1. Same-Origin

On the client-side, same-origin determination is defined in the fetch spec (https://html.spec.whatwg.org/multipage/browsers.html#origin).

On the server-side, a request with a "Sec-Fetch-Site:" request header with a value of "same-origin" is to be considered a same-origin request.

*For any request that is same-origin:

- Response **MAY** be used as a dictionary.
- Response **MAY** be compressed by an available dictionary.

### 9.3.2.2. Cross-Origin

For requests that are not same-origin *(Section 9.3.2.1)*, the "mode" of the request can be used to determine the readability of the response.

For clients that conform to the fetch spec, the mode of the request is stored in the RequestMode attribute of the request (https://fetch.spec.whatwg.org/#requestmode).

For servers responding to clients that expose the request mode information, the value of the mode is sent in the "Sec-Fetch-Mode" request header.

If a "Sec-Fetch-Mode" request header is not present, the server **SHOULD** allow for the dictionary compression to be used.

1. If the mode is "navigate" or "same-origin":

   *Response **MAY** be used as a dictionary.
   *Response **MAY** be compressed by an available dictionary.

2. If the mode is "cors":

   *For clients, apply the CORS check from the fetch spec (https://fetch.spec.whatwg.org/#cors-check) which includes credentials checking restrictions that may not be possible to check on the server.

   - If the CORS check passes:

     o Response **MAY** be used as a dictionary.
oResponse **MAY** be compressed by an available dictionary.

-Else:

  oResponse **MUST NOT** be used as a dictionary.

  oResponse **MUST NOT** be compressed by an available dictionary.

*For servers:

-If the response does not include an "Access-Control-Allow-Origin" response header:

  oResponse **MUST NOT** be used as a dictionary.

  oResponse **MUST NOT** be compressed by an available dictionary.

-If the request does not include an "Origin" request header:

  oResponse **MUST NOT** be used as a dictionary.

  oResponse **MUST NOT** be compressed by an available dictionary.

-If the value of the "Access-Control-Allow-Origin" response header is ":*":

  oResponse **MAY** be used as a dictionary.

  oResponse **MAY** be compressed by an available dictionary.

-If the value of the "Access-Control-Allow-Origin" response header matches the value of the "Origin" request header:

  oResponse **MAY** be used as a dictionary.

  oResponse **MAY** be compressed by an available dictionary.

3. If the mode is any other value (including "no-cors"):

  *Response **MUST NOT** be used as a dictionary.

  *Response **MUST NOT** be compressed by an available dictionary.
10. Privacy Considerations

Since dictionaries are advertised in future requests using the hash of the content of the dictionary, it is possible to abuse the dictionary to turn it into a tracking cookie.

To mitigate any additional tracking concerns, clients **MUST** treat dictionaries in the same way that they treat cookies. This includes partitioning the storage as cookies are partitioned as well as clearing the dictionaries whenever cookies are cleared.

11. References

11.1. Normative References


11.2. Informative References


Authors' Addresses

Patrick Meenan (editor)
Google LLC

Email: pmeenan@google.com

Yoav Weiss (editor)
Shopify Inc

Email: yoav.weiss@shopify.com