Abstract

This document specifies a new HTTP/2 frame type and new HTTP/2 settings values that would enable the use of previously transferred data as compression dictionaries, significantly improving overall compression ratio for a given connection.

In addition, this document proposes to define a set of industry standard, static, dictionaries to be used with any Lempel-Ziv based compression for the common textual MIME types prevalent on the web.

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 http://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 May 4, 2017.

Copyright Notice

Copyright (c) 2016 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 (http://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...
1. Introduction

The HTTP/2 [RFC7540] protocol encourages the use of many small assets for CSS/JS/HTML, due to its multiplexed nature. Prior to HTTP/2, asset inlining was encouraged, resulting in fewer, larger assets per website.

The HTTP/2 protocol also allows for transmitted data to be compressed with a lossless compression format. The format used is specified in the "Content-Encoding" (see [RFC2616], section 14.11) header field. For example, "Content-Encoding: br" means the data was compressed using the Brotli format.

The nature of the compression algorithms, such as DEFLATE [RFC1951] and Brotli [RFC7932], used with HTTP in practice, require a certain "window" of data to perform backward matching. Therefore, larger files have much better compression ratio. To improve compression for smaller files, these algorithms allow to use a chunk of arbitrary data as a "Custom Dictionary" and function as the initial sliding window.

While compression, especially of dynamic resources, is a compute-heavy operation, where investing more compute power results in diminishing returns (in terms of compression ratio). This technique is known to improve compression ratio significantly, while not
requiring significant additional processing time, and is supported by most LZ based compression formats.

This document introduces a mechanism for using previously transmitted data over HTTP/2 as a dictionary to be used with an underlying compression algorithm.

1.1. Conventions and Terminology

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 [RFC 2119].

2. HTTP/2 Extension

2.1. Extension Settings

The extension introduces two new SETTINGS values.

SETTINGS_ENABLE_DICTIONARIES(0xTBA): This setting can be used to enable the use of Compression Dictionaries for a given connection. The value indicates how many dictionaries the sender is willing to maintain. The default value is 0, the maximal value is 256. Value higher than 256 may indicate that the sender is willing to initialize the dictionaries with the default preset values. In that case the actual value for this setting is computed as "value - 256".

SETTINGS_MAX_DICTIONARY_SIZE(0xTBA): Indicates the size of the largest dictionary that the sender is willing to maintain, in octets. The initial value is $2^{17}$ (131,072 octets).

2.2. Extension Frames

2.2.1. The SET_DICTIONARY Frame

The SET_DICTIONARY frame (type=0xTBA).

+-------------+-------------+
| Dict ID (8) |   Size (8)  |
+-------------+-------------+

The SET_DICTIONARY frame can be sent from the server to the client, on any client initiated stream in the open or half-closed (remote) states. The SET_DICTIONARY frame MUST precede any DATA frames on that stream. The SET_DICTIONARY frame SHOULD be followed by sufficient DATA frames to fill Size octets after decompression, even if an RST
frame was received for that stream. If not enough DATA was sent, the Dictionary for the given ID is considered uninitialized.

More than one SET_DICTIONARY frames MAY be sent on a given stream.

The SET_DICTIONARY frame contains the following fields:

Dict ID: an 8-bit ID, indicates the dictionary. MUST be lower than the value agreed by the SETTINGS_ENABLE_DICTIONARIES setting.

Size: Indicates how many octets (as a power of 2) of the given stream will be used. If Size is greater than the length of the transmitted data, then all of the data will be used.

The SET_DICTIONARY frame defines the following flag:

APPEND (0x1): Indicates that the data is to be appended to the existing dictionary with the given ID, as opposed to replacing it with the new data.

2.2.2. The USE_DICTIONARY Frame

The USE_DICTIONARY frame (type=0xTBA).

```
+-------------+
| Dict ID (8) |
+-------------+
```

The USE_DICTIONARY frame indicates that the current frame is to be decompressed with the indicated dictionary. The USE_DICTIONARY frame MUST be sent before any DATA frame on a given stream. SET_DICTIONARY and USE_DICTIONARY frames can be sent on the same stream.

The USE_DICTIONARY frame contains the following fields:

Dict ID: an 8-bit ID that indicates which dictionary to use.

2.3. Static Dictionaries

This document proposes to generate a set of up to 8 standard dictionaries to be optionally bundled with supporting implementations. Each dictionary should be 32,768 or 65,536 octets long.

Each static dictionary will be identified by an integer ID in the range {1..7}.  

A client that supports the use of static dictionaries, will notify the server by sending a value of SETTINGS_ENABLE_DICTIONARIES greater than 256.

If the server also supports the use of static dictionaries, it will respond in a similar fashion.

If the client and the server agreed on the use of static dictionaries, then both will initialize the first 8 dictionaries, with the contents of the static dictionaries. If the value of SETTINGS_ENABLE_DICTIONARIES agreed upon is lower than 256 + 8, then as many dictionaries as needed will be initialized.

### 3. Dictionary State

Both the server and the client MUST process the SET_DICTIONARY and USE_DICTIONARY frames in the order they are sent/received, with the exception when both are sent over the same stream. In that case USE_DICTIONARY is processed prior to all the SET_DICTIONARY frames.

Doing otherwise will result in an illegal state of the dictionaries. This is similar to the way HEADER frames are processed in order to maintain legal HPACK state on the server and the client.

All the dictionaries are initially uninitialized. If the use of static dictionaries is agreed upon by both parties, then the dictionaries are initialized as described in Section 2.3.

When SET_DICTIONARY is used, both the server and the client will either use the first Size bytes of the stream, after decompression, as the dictionary, or in case the APPEND flag is set, take the first Size of the stream after decompression, and append it from the end of the existing dictionary. If the resulting dictionary is larger than the SETTINGS_MAX_DICTIONARY_SIZE, then only the last SETTINGS_MAX_DICTIONARY_SIZE octets are used.

### 3.1. Server Behaviour

The server MAY send a SET_DICTIONARY frame on any client initiated stream in the open or half-closed (remote) states, prior to sending any DATA on that stream.

After the server sends a SET_DICTIONARY stream with a given ID, it MUST not use the dictionary for compression, until it sent sufficient data for the dictionary to become usable by the client. Sufficient data is computed as the size of the data after decompression.
After sufficient data was sent, the server MAY use the associated dictionary on any subsequent stream by sending the USE_DICTIONARY frame.

3.2. Client Behaviour

When receiving a USE_DICTIONARY frame, the client will use the specified dictionary to decompress the DATA.

A given stream may receive a SET_DICTIONARY and USE_DICTIONARY with the same ID. In that case the stream is decompressed with the old dictionary and then the dictionary is updated as required.

If a USE_DICTIONARY frame arrives for an uninitialized dictionary, this is considered as stream error of type COMPRESSION_ERROR.

4. Security Considerations

As with any compression scheme, using cross-stream compression is potentially more sensitive to [BREACH] type of attacks.

Therefore, this extension SHALL be disabled by default by all server implementations.

If a server acts as an intermediary, then it MUST NOT enable cross-stream compression, unless the origin also enables cross-stream compression. If the origin does not use the HTTP/2 protocol, the intermediary server MUST NOT use cross-stream compression, unless notified by the origin explicitly by either a receipt of a pre-agreed upon HTTP header, or out-of-band.

In addition, a server SHOULD avoid the use cross-stream compression on cross-site requests.

The serve MAY always use the predefined static dictionaries.

5. References

5.1. Normative References

5.2. Informative References


Author’s Address

Vlad Krasnov
Cloudflare Inc.

Email: vlad@cloudflare.com