Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 8, 2021

M. Nottingham
July 7, 2020

Greasing HTTP
draft-nottingham-http-grease-00

Abstract

Like many network protocols, HTTP is vulnerable to ossification of its extensibility points. This draft specifies how they should be exercised ('greased') to assure their continued usability.

Note to Readers

RFC EDITOR: please remove this section before publication

The issues list for this draft can be found at https://github.com/mnot/I-D/labels/http-grease [1].

The most recent (often, unpublished) draft is at https://mnot.github.io/I-D/http-grease/ [2].

Recent changes are listed at https://github.com/mnot/I-D/commits/gh-pages/http-grease [3].

See also the draft's current status in the IETF datatracker, at https://datatracker.ietf.org/doc/draft-nottingham-http-grease/ [4].

Status of This Memo

Table of Contents

1. Introduction

Like many network protocols, HTTP is vulnerable to ossification of its extensibility points. Those that are rarely exercised risk 'rusting shut' because recipients assume that they will not be used. This happens in practice for several reasons, including implementer convenience, performance optimisation, or traffic characterisation.

Because extensibility is a primary mechanism for protocol evolution, it is important to keep these extension points flexible. For points that are rarely used, one proven way (pioneered by [RFC8701]) to

assure this is through sending 'grease' values - i.e., extension values that are hard to predict and have no effect on correct protocol operation.

This document specifies how HTTP's extension points should be greased, to assure their continued usability. It focuses on generic HTTP features; other documents cover versioned extensibility points (e.g., see [I-D.bishop-httpbis-grease]).

1.1. What to Grease?

HTTP has several extension mechanisms. While keeping all of them available for use is desirable, this document currently targets two specific extensibility points - HTTP request header fields and request cache directives - for a few reasons.

Some extension points are not practical to grease. For example, introducing new HTTP methods is important, but greasing them would require sending requests beyond those intended by the user. Beyond the overhead of doing so, failure of those requests is not likely to create an incentive to allow those requests, because that failure is not user visible.

Other extension points are already effectively ossified: for example, range units. While it might be possible to introduce a new range unit in the future (with enough effort), there is not much desire to do so in the community at this point, and the risk of greasing it causing too many failures is high.

Greasing aspects of HTTP responses (such as header fields or cache control directives) is not addressed in this document because current Web traffic already effectively greases them. For example, the breadth of unrecognised headers sent from HTTP servers effectively keeps response header fields greased; likewise with response cache-control directives.

Future revisions might address other extensibility points (including those listed above), based upon discussion and feedback.

1.2. How to Grease?

Greasing has the goal of keeping protocol extension points flexible - that is, it should remain possible to introduce new values with negligible risk of interoperability problems. By necessity, this is not absolute; an implementation determined to control input values can anticipate grease values and allow them, while denying other extensions.

As a result, one of the tradeoffs in a greasing mechanism is between making the values difficult to anticipate and the complexity of the mechanism. One that is hard to anticipate typically requires hard-to-predict values generated by an algorithm, with a corresponding

prohibition on registration of those values. Even then, a determined implementation could use heuristics to identify and allow grease values, while blocking others. On the other hand, an easily predictable value can be added to an allow list in implementations, while they still block unknown values.

This document's initial goal is to make it possible to deploy new standards-defined values with a suitable notice period, rather than to allow any implementation to introduce new values at any time. To meet that goal, a 'HTTP grease process' is defined, whereby grease values are periodically announced and later sent by implementations, so that receiving implementations have enough time to assure interoperability.

1.3. 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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. The HTTP Grease Process

There are a few parties involved in the greasing of an HTTP extensibility point. They are:

- Grease senders - implementers and operators of HTTP deployments that send grease extensions

- Grease recipients - implementers and operators of HTTP deployments that receive grease extensions

- The grease coordinator - a person appointed to oversee the HTTP grease process

To aid communication between these parties, a mailing list (TBD) has been created for announcements and discussion.

The grease coordinator is appointed by the ART Area Director(s), in consultation with the HTTP Working Group chair(s).

On a periodic basis (to be determined by the grease coordinator in consultation with grease senders and grease recipients), the grease coordinator will announce a new grease value for the extension points covered by this process. Where possible, these values will be provisionally registered with IANA, with the note 'grease value' and a reference to this specification.

Such announcements MUST contain the details of the grease value (see individual requirements below), and a date on which grease senders SHOULD start sending that value.

Grease senders SHOULD send the grease value on a sizeable fraction of traffic (e.g., 1/8th); too small a proportion might be ignored.

Grease senders MAY selectively send grease values. For example, a Web browser might only send grease on navigation requests, to assure that any interoperability problems are clearly visible.

Grease senders SHOULD NOT send grease values when the request method is non-idempotent or unsafe.

If grease senders or recipients experience widespread interoperability problems as the result of deployment of a grease value, they SHOULD report this to the grease coordinator, who MAY declare that the grease value can be withdrawn by grease senders. Grease senders SHOULD NOT act unilaterally in such cases.

Once a new grease value has been deployed, old grease values SHOULD be withdrawn by grease senders.

2.1. Greasing HTTP Request Header Fields

Grease values for HTTP request header fields consist of a field name and a field value.

Grease field names SHOULD be hard to predict; e.g., they SHOULD NOT have any identifying prefix, suffix, or pattern. However, they MUST NOT be likely to conflict with unregistered or future field names, and the grease coordinator MUST avoid potentially offensive or confusing terms. They also MUST conform to the syntactic requirements for field names in HTTP ([I-D.ietf-httpbis-semantics], Section 4.3).

This can be achieved in different ways (which SHOULD vary from time to time), for example:

- Combine two or three dictionary words or proper nouns with a hyphen (e.g., 'Skateboard-Clancy', 'Murray-Fortnight-Scout')
- Append digits to a dictionary word (e.g., 'Turnstile23')
- Generate a string using a hash or similar function (e.g., 'dd722785c01b')

Grease field values can be statically specified in the grease announcement, specified to be of a certain type (e.g., using [I-D.ietf-httpbis-header-structure] types), or left to the discretion of the grease sender.

2.2. Greasing HTTP Request Cache Directives

Grease values for HTTP request cache directives consist of a directive name and an optional directive value.

Grease directive names SHOULD be hard to predict; e.g., they SHOULD NOT have any identifying prefix, suffix, or pattern. However, they MUST NOT be likely to conflict with unregistered or future directive names, and the grease coordinator MUST avoid potentially offensive or confusing terms. They also MUST conform to the syntactic

requirements for cache directive names in HTTP ([I-D.ietf-httpbis-cache], Section 5.2).

This can be achieved in different ways (which SHOULD vary from time to time), for example:

- Select a single dictionary word or proper noun (e.g., 'fanciful', 'imagine')
- Combine two dictionary words or proper nouns with a hyphen (e.g., 'skateboard-clancy')
- Append digits to a dictionary word (e.g., 'turnstile23')
- Generate a string using a hash or similar function (e.g., 'dd722785c01b')

Grease field values can be omitted (so there is no '=value'), statically specified in the grease announcement, specified to be of a certain type (e.g., an integer, a quoted string), or left to the discretion of the grease sender.

3. Security Considerations

Some HTTP extensibility points are becoming (or have become) ossified because of security considerations; receiving implementations believe that it is more secure to reject unknown values, or they are able to identify undesirable peers through their use of extensions.

This document does not directly address these concerns, nor does it directly disallow such behaviour. Instead, it aims to encourage the ability to accommodate new extensions more quickly than is now currently possible.

4. References

4.1. Normative References

[I-D.ietf-httpbis-cache]
Fielding, R., Nottingham, M., and J. Reschke, "HTTP Caching", draft-ietf-httpbis-cache-08 (work in progress), May 2020.

[I-D.ietf-httpbis-semantics]
Fielding, R., Nottingham, M., and J. Reschke, "HTTP Semantics", draft-ietf-httpbis-semantics-08 (work in progress), May 2020.

[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/info/rfc2119>.

[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/info/rfc8174>.

4.2. Informative References

[I-D.bishop-httpbis-grease]
Bishop, M., "GREASE for HTTP/2", draft-bishop-httpbis-grease-01 (work in progress), June 2020.

[I-D.ietf-httpbis-header-structure]
Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", draft-ietf-httpbis-header-structure-19 (work in progress), June 2020.

[RFC8701] Benjamin, D., "Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility", RFC 8701, DOI 10.17487/RFC8701, January 2020, <https://www.rfc-editor.org/info/rfc8701>.

4.3. URIs

[1] https://github.com/mnot/I-D/labels/http-grease

[2] https://mnot.github.io/I-D/http-grease/

[3] https://github.com/mnot/I-D/commits/gh-pages/http-grease

[4] https://datatracker.ietf.org/doc/draft-nottingham-http-grease/

Appendix A. Bootstrapping the HTTP Grease Process

Because the initial focus of this document is on two request extension points, the relevant grease senders will be HTTP clients - a combination of Web browsers, HTTP client libraries and intermediaries (such as CDNs). The relevant grease recipients will be HTTP servers (both on the origin and in intermediaries).

Broadly speaking, HTTP servers accept these extensions, unless they have a Web Application Firewall (WAF) installed. As such, greasing HTTP successfully will require client implementers, WAF vendors, and in some cases WAF deployers to work together.

Clients are likely to be risk-averse; if their implementation alone breaks some Web sites, they can face negative consequences (because their users can easily flee to other implementations). Therefore, a successful greasing strategy needs to include most or all major clients, and their actions need to be coordinated.

WAF vendors and deployers often do not coordinate behaviour, and may not have prompt update mechanisms. Therefore, a successful greasing strategy needs to attract them to into community engagement (e.g., using the mailing list above) and needs to accommodate their needs; it is likely they will not be able to deploy updates quickly at first, for example.

As a result, when greasing begins, it will be necessary to have long lead times between announcement and sending. Likewise, initial grease values are more likely to succeed (building confidence and engagement) if they are static and simple.

For example, the first grease value might be completely static, very simple (e.g., "Grease: 1"), and announced several months ahead of time. Subsequent values can grow in complexity, become more dynamic, and arrive with progressively shorter notice, after discussion within the community.

Some clients may not be able to deploy new grease values on a regular basis, and so they will need some sort of update or scheduling mechanism to participate.

In cases where greasing breaks deployed sites too widely, clients may wish to temporarily stop greasing while the issue is mitigated. This should be coordinated among all clients, rather than done unilaterally. Mitigations like retrying requests without grease can be performed at any time; the point is to gently increase pressure on servers to accept new values, not to break sites unnecessarily.

Author's Address

Mark Nottingham

Email: mnot@mnot.net
URI: https://www.mnot.net/