Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 21, 2012

J. Snell
December 19, 2011

Prefer Header for HTTP
draft-snell-http-prefer-10

Abstract

This specification defines an HTTP header field that can be used by a client to request that certain behaviors be implemented by a server while processing a request.

Status of this Memo

This Internet-Draft is submitted to IETF 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 June 21, 2012.

Table of Contents

1. Introduction

This specification defines a new HTTP request header field that may be used by clients to request optional behaviors be applied by a server during the processing the request.

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

1.1. Syntax Notation

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234] and includes, by reference, the "token", "quoted-string", "OWS", "BWS" rules and the #rule extension as defined within Section 1.2 [I-D.ietf-httpbis-p1-messaging].

2. The Prefer Request Header

The Prefer request-header field is used to indicate that particular server behaviors are preferred by the client, but not required for successful completion of the request. Prefer is similar in nature to the Expect header field defined by Section 9.3 of [I-D.ietf-httpbis-p2-semantics] with the exception that servers are allowed to ignore stated preferences.

```
Prefer     = "Prefer" ":" 1#preference
preference = token [ BWS "=" BWS value ]
             *( OWS ";" [ OWS parameter ] )
parameter  = token [ BWS "=" BWS value ]
value      = token / quoted-string
```

This header field is defined with an extensible syntax to allow for future values included in the Registry of Preferences (Section 9.1)). A server that does not recognize or is unable to comply with particular preference tokens in the Prefer header field of a request MUST ignore those tokens and MUST NOT stop processing or signal an error.

A preference token MAY specify a value. Empty, or zero length values on both the preference token and within parameters are equivalent to no value being specified at all. The following, then, are equivalent:

```
Prefer: foo; bar
Prefer: foo; bar=""
Prefer: foo=""; bar
```

An optional, arbitrary collection of parameters MAY be specified for any preference token. The meaning and application of such parameters is dependent on the definition of each preference token and the server's implementation thereof.

If a particular preference token or parameter is specified multiple times, repeated occurrences MUST be ignored without signaling an error or otherwise altering the processing of the request.

Comparison of preference token names is case-insensitive while values are case-sensitive regardless of whether token or quoted-string values are used.

The Prefer request header field MUST be forwarded by a proxy if the request is forwarded. In various situations, A proxy may determine that it is capable of honoring a preference independently of the server to which the request is directed. For instance, an intervening proxy may be capable of transparently providing asynchronous handling of a request using a 202 Accepted responses independently of the origin server. Such proxies could choose to honor the "return-asynch" preference. Individual preference tokens MAY define their own requirements and restrictions as to whether and how proxies may apply the preference to a request independently of the origin server.

As per Section 3.2 of [I-D.ietf-httpbis-p1-messaging], Implementations MUST be capable of supporting either multiple instances of the Prefer header field in a single message as well as multiple preference tokens separated by commas in a single Prefer header, for instance, the following examples are equivalent:

Multiple Prefer Header Fields:

```
POST /foo HTTP/1.1
Host: example.org
Prefer: return-asynch
Prefer: wait=100
Date: Tue, 20 Dec 2011 12:34:56 GMT
```

Single Prefer Header Field:

```
POST /foo HTTP/1.1
Host: example.org
Prefer: return-asynch, wait=100
Date: Tue, 20 Dec 2011 12:34:56 GMT
```

2.1. Content Negotiation and Cache Considerations

Note that while the Prefer header field is not intended to be used as content negotiation mechanism, the application of a preference potentially could affect the caching characteristics of a response. Specifically, if a server supports the optional application of a preference that could even potentially result in a variance to a cache's handling of a response entity, a Vary header field MUST be included with the response listing the Prefer header field regardless of whether the client actually uses Prefer in the request.

Because of the inherent complexities involved with properly implementing server-driven content negotiation, effective caching, and the application of optional preferences, implementors must exercise caution when utilizing preferences in such a way as to impact the caching of a response and SHOULD avoid using the Prefer header mechanism for content negotiation.

2.2. Examples

The following examples illustrate the use of various Preferences defined by this specification, as well as undefined extensions for strictly illustrative purposes:

Return a 202 Accepted response for asynchronous processing if the response cannot be processed within 10 seconds. An undefined "priority" preference is also specified.

```
Prefer: return-asynch, wait=10;
Prefer: priority=5;
```

Use lenient processing

```
Prefer: Lenient
```

Use of an optional, undefined parameter on the return-minimal preference requesting a response status code of 204 for a successful response.

```
Prefer: return-minimal; status=204
```

3. The "return-asynch" Preference

The "return-asynch" preference indicates that the client prefers the server to respond asynchronously to a response. For instance, in the case when the length of time it takes to generate a response will exceed some arbitrary threshold established by the server, the server

may honor the return-asynch preference by returning either a 202 Accepted or 303 See Other response.

```
return-asynch = "return-asynch"
```

The key motivation for the "return-asynch" preference is to facilitate the operation of asynchronous request handling by allowing the client to indicate to a server it's capability and preference for handling asynchronous responses.

An example request specifying the "return-asynch" preference:

```
POST /collection HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: return-asynch

{Data}
```

An example asynchronous response using 202 Accepted:

```
HTTP/1.1 202 Accepted
Location: http://example.org/collection/123
```

An alternative asynchronous response using 303 See Other:

```
HTTP/1.1 303 See Other
Location: http://example.org/collection/123
Retry-After: 10
```

4. The "return-representation" Preference

The "return-representation" preference indicates that the client prefers that the server include an entity representing the current state of the resource in the response to a successful request.

```
return-representation = "return-representation"
```

When honoring the "return-representation" preference, the server MUST include a Content-Location header field specifying the URI of the resource representation being returned. Per section 6.1 of [I-D.ietf-httpbis-p2-semantics], the presence of the Content-Location header field in the response asserts that the payload is a representation of the resource identified by the Content-Location URI.

The "return-representation" preference is intended primarily to

provide a means of optimizing communication between the client and server by eliminating the need for a subsequent GET request to retrieve the current representation of the resource following a modification.

Currently, after successfully processing a modification request such as a POST or PUT, a server may choose to return either an entity describing the status of the operation or a representation of the modified resource itself. While the selection of which type of entity to return, if any at all, is solely at the discretion of the server, the "return-representation" preference -- along with the "return-minimal" preference defined below -- allow the server to take the client's preferences into consideration while constructing the response.

An example request specifying the "return-representation" preference:

```
PUT /collection/123 HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: return-representation

{Data}
```

An example response containing the resource representation:

```
HTTP/1.1 200 OK
Content-Location: http://example.org/collection/123
Content-Type: text/plain

{Data}
```

5. The "return-minimal" Preference

The "return-minimal" preference indicates that the client wishes the server to return a minimal response to a successful request. Typically, such responses would utilize the 204 No Content status, but other codes MAY be used as appropriate, such as a 200 status with a zero-length response entity. The determination of what constitutes an appropriate minimal response is solely at the discretion of the server.

```
return-minimal = "return-minimal"
```

The "return-minimal" preference is intended to provide a means of optimizing communication between the client and server by reducing the amount of data the server is required to return to the client

following a request. This can be particularly useful, for instance, when communicating with limited-bandwidth mobile devices or when the client simply does not require any further information about the result of a request beyond knowing if it was successfully processed.

An example request specifying the "return-minimal" preference:

```
POST /collection HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: return-minimal

{Data}
```

An example response containing the resource representation:

```
HTTP/1.1 201 Created
Location: http://example.org/collection/123
Content-Length: 0
```

6. The "wait" Preference

The "wait" preference can be used to establish an upper bound on the length of time, in seconds, the client is willing to wait for a response, after which the client may choose to abandon the request. In the case generating a response will take longer than the time specified, the server, or proxy, MAY choose to utilize an asynchronous processing model by returning, for example, 202 Accepted or 303 See Other responses.

```
wait = "wait" BWS "=" BWS delta-seconds
```

Clients specifying the "wait" Preference SHOULD also use the Date header field, as specified in Section 9.2 of [I-D.ietf-httpbis-p2-semantics], within the request to establish the time at which the client began waiting for the completion of the request. Failing to include a Date header field in the request would require the server to use the instant it received or began processing the request as the baseline for determining how long the client has been waiting which could yield unintended results.

The lack of a Date header in the request, or poor clock synchronization between the client and server makes it impossible to determine the exact length of time the client has already been waiting when the request is received by the server. The only reliable information conveyed by the wait preference is that the client is not expecting the server to spend more than the specified

time on request processing and may terminate the transaction at any time.

An example request specifying the "wait" and "return-asynch" preferences to indicate that the client wishes the server to respond asynchronously if processing of the request will take longer than 10 seconds:

```
POST /collection HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: return-asynch, wait=10
Date: Tue, 20 Dec 2011 12:34:56 GMT

{Data}
```

7. The "strict" and "lenient" Processing Preferences

The "strict" and "lenient" preferences are mutually-exclusive directives indicating, at the servers discretion, how the client wishes the server to handle potential error conditions that may arise in the processing of a request. For instance, if the payload of a request contains various minor syntactical or semantic errors, but the server is still capable of comprehending and successfully processing the request, a decision must be made to either reject the request with an appropriate 4xx error response or to go ahead with processing. The "strict" preference can be used by the client to indicate that, in such conditions, it would prefer that the server reject the request, while the "lenient" preference indicates that the client would prefer the server to attempt to process the request. The specific meaning and application of the "strict" and "lenient" directives is specific to each type of resource, the request method and the operation of the server.

```
handling = "strict" / "lenient"
```

An example request specifying the "strict" preference:

```
POST /collection HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: strict
```

An example request specifying the "lenient" preference:

```
POST /collection HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: lenient
```

8. Registered Preferences

Well-defined preferences can be registered for convenience and/or to promote reuse by other applications. This specification establishes an IANA registry of such relation types see Section 9.1.

Registered preference names MUST conform to the token rule, and MUST be compared character-by-character in a case-insensitive fashion. They SHOULD be appropriate to the specificity of the preference; i.e., if the semantics are highly specific to a particular application, the name should reflect that, so that more general names are available for less specific use.

Registered preferences MUST NOT constrain servers, clients or any intermediaries involved in the exchange and processing of a request to any behavior required for successful processing. The use and application of a preference within a given request MUST be optional on the part of all participants.

9. IANA Considerations

The 'Prefer' header field should be added to the permanent registry (see [RFC3864]).

```
Header field name: Prefer
Applicable Protocol: HTTP
Status:
Author: James M Snell <jasnell@gmail.com>
Change controller: IETF
Specification document: this specification
```

9.1. The Registry of Preferences

Preferences are registered on the advice of a Designated Expert (appointed by the IESG or their delegate), with a Specification Required (using terminology from [RFC5226]).

The requirements for registered preferences are described in Section 8.

Registration requests consist of the completed registration template below, typically published in an RFC or Open Standard (in the sense described by Section 7 of [RFC2026]). However, to allow for the allocation of values prior to publication, the Designated Expert may approve registration once they are satisfied that a specification will be published.

Note that relation types can be registered by third parties, if the Designated Expert determines that an unregistered relation type is widely deployed and not likely to be registered in a timely manner.

The registration template is:

- Preference: (A value for the Prefer request header field that conforms to the syntax rule given in Section 2)
- Description:
- Reference:
- Notes: [optional]
- Application Data: [optional]

Registration requests should be sent to the preferences@ietf.org mailing list, marked clearly in the subject line (e.g., "NEW PREFERENCE - example" to register an "example" preference).

Within at most 14 days of the request, the Designated Expert(s) will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful.

Decisions (or lack thereof) made by the Designated Expert can be first appealed to Application Area Directors (contactable using app-ads@tools.ietf.org email address or directly by looking up their email addresses on http://www.iesg.org/ website) and, if the appellant is not satisfied with the response, to the full IESG (using the iesg@iesg.org mailing list).

IANA should only accept registry updates from the Designated Expert(s), and should direct all requests for registration to the review mailing list.

9.1.1. Initial Registry Contents

The Preferences Registry's initial contents are:

- Preference: return-asynch

- Description: Indicates that the client prefers the server to respond asynchronously to a request as described by Section 3
- Reference: [this specification]

- Preference: return-minimal
- Description: Indicates that the client prefers the server return a minimal response to a request as described by Section 5
- Reference: [this specification]

- Preference: return-representation
- Description: Indicates that the client prefers the server to include a representation of the current state of the resource in response to a request as described by Section 4
- Reference: [this specification]

- Preference: wait
- Description: Indicates an upper bound to the lenght of time the client is willing to wait for a response, after which the request may be aborted.
- Reference: [this specification]

- Preference: strict
- Description: Indicates that the client wishes the server to apply strict validation and error handling to the processing of a request.
- Reference: [this specification]

- Preference: lenient
- Description: Indicates that the client wishes the server to apply lenient validation and error handling to the processing of a request.
- Reference: [this specification]

10. Security Considerations

Specific preferences requested by a client can introduce security considerations and concerns beyond those discussed in HTTP/1.1 Parts 1 [I-D.ietf-httpbis-p1-messaging], 2 [I-D.ietf-httpbis-p2-semantics], 3 [I-D.ietf-httpbis-p3-payload], 4 [I-D.ietf-httpbis-p4-conditional], 5 [I-D.ietf-httpbis-p5-range], 6 [I-D.ietf-httpbis-p6-cache], and 7 [I-D.ietf-httpbis-p7-auth]. Implementors must refer to the specifications and descriptions of each preference to determine the security considerations relevant to each.

11. Normative References

[I-D.ietf-httpbis-p1-messaging]
Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., and J. Reschke, "HTTP/1.1, part 1: URIs, Connections, and Message Parsing", draft-ietf-httpbis-p1-messaging-17 (work in progress), October 2011.

[I-D.ietf-httpbis-p2-semantics]
Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., and J. Reschke, "HTTP/1.1, part 2: Message Semantics", draft-ietf-httpbis-p2-semantics-17 (work in progress), October 2011.

[I-D.ietf-httpbis-p3-payload]
Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., and J. Reschke, "HTTP/1.1, part 3: Message Payload and Content Negotiation", draft-ietf-httpbis-p3-payload-17 (work in progress), October 2011.

[I-D.ietf-httpbis-p4-conditional]
Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., and J. Reschke, "HTTP/1.1, part 4: Conditional Requests", draft-ietf-httpbis-p4-conditional-17 (work in progress), October 2011.

[I-D.ietf-httpbis-p5-range]
Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., and J. Reschke, "HTTP/1.1, part 5: Range Requests and Partial Responses", draft-ietf-httpbis-p5-range-17 (work in progress), October 2011.

[I-D.ietf-httpbis-p6-cache]
Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., Nottingham, M., and J. Reschke, "HTTP/1.1, part 6: Caching", draft-ietf-httpbis-p6-cache-17 (work in progress), October 2011.

[I-D.ietf-httpbis-p7-auth]
Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T., Lafon, Y., and J. Reschke, "HTTP/1.1, part 7: Authentication",

draft-ietf-httpbis-p7-auth-17 (work in progress), October 2011.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, September 2004.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

Author's Address

James M Snell

Email: jasnell@gmail.com