Network Working Group J. Snell
Internet-Draft August 23, 2012
Intended status: Standards Track
Expires: February 24, 2013

Prefer Header for HTTP
draft-snell-http-prefer-14

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

Copyright Notice

Table of Contents

1. Introduction

Within the course of processing an HTTP request there are typically a range of required and optional behaviors that a server or intermediary can employ. These often manifest is a variety of subtle and not-so-subtle ways within the response.

For example, when using the HTTP PUT method to modify a resource -- similar to that defined for the Atom Publishing Protocol [RFC 5023] -- the server is given the option of returning either a complete representation of a modified resource or a minimal response that indicates only the successful completion of the operation. The selection of which type of response to return to the client generally has no bearing on the successful processing of the request but could, for instance, have an impact on what actions the client must take after receiving the response. That is, returning a representation of the modified resource within the response can allow the client to avoid sending an additional subsequent GET request.

Similarly, servers that process requests are often faced with decisions about how to process requests that may be technically invalid or incorrect but are still understandable. It might be the case that the server is able to overlook the technical errors in the request but still successfully process the request. Depending on the specific requirements of the application and the nature of the request being made, the client might or might not consider such lenient processing of its request to be appropriate.

While the decision of exactly which behaviors to apply in these cases lies with the server processing the request, the server might wish to defer to the client to specify which optional behavior is preferred.

Currently, HTTP offers no explicitly defined means of expressing the client's preferences regarding the optional aspects of handling of a given request. While HTTP does provide the Expect header -- which can be used to identify mandatory expectations for the processing of a request -- use of the field to communicate optional preferences is problematic:

1. The semantics of the Expect header field are such that intermediaries and servers are required to reject any request that states unrecognized or unsupported expectations.
2. While the Expect header field is end-to-end, the HTTP specification requires that the header be processed hop-by-hop. That is, every interceding intermediary that handles a request between the client and the origin server is required to process an expectation and determine whether it is capable of appropriately handling it.

The rigid, must-understand semantics of the Expect header, therefore, make it a poor choice for the general expression of optional preferences that may be specific to an individual application and are therefore unknown to an intermediary or are otherwise irrelevant to the intermediaries successful handling of the request and response.

Another option available to clients is to utilize Request URI query-string parameters to express preferences. Doing so, however, results in a variety of issues affecting the cacheability of responses.

As an alternative, this specification defines a new HTTP request header field that can be used by clients to request that optional behaviors be applied by a server during the processing the request. Additionally, a handful of initial preference tokens for use with the new header are defined.

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", "word", "OWS", "BWS" rules and the #rule extension as defined within Sections 1.2 and 3.2.4 of [I-D.ietf-httpbis-p1-messaging].

2. The Prefer Request Header Field

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 word ]
               *( OWS ";" [ OWS parameter ] )
  parameter  = token [ BWS "=" BWS word ]
```

This header field is defined with an extensible syntax to allow for future values included in the Registry of Preferences (Section 4.1). 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 can contain 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 set of parameters can 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 is end-to-end and MUST be forwarded by a proxy if the request is forwarded.

In various situations, a proxy might determine that it is capable of honoring a preference independently of the server to which the request has been directed. For instance, an intervening proxy might be capable of providing asynchronous handling of a request using 202 Accepted responses independently of the origin server. Such proxies can choose to honor the "return-asynch" preference on their own despite whether the origin is capable or willing to do so. In such cases, however, the proxy is still required to forward the Prefer header on to the origin server.

Individual preference tokens MAY define their own requirements and restrictions as to whether and how intermediaries can 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 support 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 field. 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: wait=100, return-asynch
  Date: Tue, 20 Dec 2011 12:34:56 GMT
```

No significance is given to the order in which preference tokens appear within a request.

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 just 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 used 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 NOT use 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:

1. 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;
```

2. Use lenient processing:

```
Prefer: Lenient
```

3. 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. Preference Definitions

The following subsections define an initial set of preferences. Additional 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 4.1).

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 remain 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.

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

ABNF:

```
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 its 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
```

3.2. 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.

ABNF:

```
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 can 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
ETag: "d3b07384d113edec49eaa6238ad5ff00"

{Data}
```

The "return-minimal" and "return-representation" preferences are mutually exclusive directives that MUST NOT be used in combination within a single request. If a server receives a request containing both the "return-minimal" and "return-representation" preferences, it MAY choose to ignore either or both of the stated preferences but MUST NOT signal an error or fail to process the request solely on the basis of those preferences.

3.3. 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.

ABNF:

```
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 minimal response:

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

The "return-minimal" and "return-representation" preferences are mutually exclusive directives that MUST NOT be used in combination within a single request. If a server receives a request containing both the "return-minimal" and "return-representation" preferences, it MAY choose to ignore either or both of the stated preferences but MUST NOT signal an error or fail to process the request solely on the basis of those preferences.

3.4. 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 might 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.

ABNF:

```
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 can 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}
```

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

The "strict" and "lenient" preferences are mutually-exclusive directives indicating, at the server's discretion, how the client wishes the server to handle potential error conditions that can 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 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.

ABNF:

```
  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
```

4. IANA Considerations

The 'Prefer' header field should be added to the Permanent Message Header Fields registry defined in [RFC3864] (http://www.iana.org/assignments/message-headers/perm-headers.html).

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

4.1. The Registry of Preferences

IANA is asked to create a new registry, "HTTP Preferences", under the Hypertext Transfer Protocol (HTTP) Parameters group. New registrations will use the Specification Required policy [RFC5226].

The requirements for registered preferences are described in Section 3.

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 can approve registration once they are satisfied that a specification will be published.

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

The registration template is:

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

Registration requests should be sent to the ietf-http-wg@w3.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.

4.2. Initial Registry Contents

The Preferences Registry's initial contents are:

- o Preference: return-asynch
- o Description: Indicates that the client prefers the server to respond asynchronously to a request.
- o Reference: [this specification], Section 3.1

- o Preference: return-minimal
- o Description: Indicates that the client prefers the server return a minimal response to a request.
- o Reference: [this specification], Section 3.3

- o Preference: return-representation
- o Description: Indicates that the client prefers the server to include a representation of the current state of the resource in response to a request.
- o Reference: [this specification], Section 3.2

- o Preference: wait
- o Description: Indicates an upper bound to the lenght of time the client is willing to wait for a response, after which the request can be aborted.
- o Reference: [this specification], Section 3.4

- o Preference: strict
- o Description: Indicates that the client wishes the server to apply strict validation and error handling to the processing of a request.

- o Reference: [this specification], Section 3.5

- o Preference: lenient
- o Description: Indicates that the client wishes the server to apply lenient validation and error handling to the processing of a request.
- o Reference: [this specification], Section 3.5

5. 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.

A server could incur greater costs in attempting to comply with a particular preference (for instance, the cost of providing a representation in a response that would not ordinarily contain one; or the commitment of resources necessary to track state for an asynchronous response). Unconditional compliance from a server could allow the use of preferences for denial of service. A server can ignore an expressed preference to avoid expending resources that it does not wish to commit.

6. 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-18 (work in progress), January 2012.

[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-18 (work in progress), January 2012.

[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-18 (work in
              progress), January 2012.

   [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-18 (work in progress),
              January 2012.

   [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-18 (work in
              progress), January 2012.

   [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-18 (work in
              progress), January 2012.

   [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-18 (work in progress),
              January 2012.

   [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