Internet Engineering Task Force (IETF)
Request for Comments: 7240
Category: Standards Track
ISSN: 2070-1721

J. Snell
June 2014

# Prefer Header for HTTP

## Abstract

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

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741[1].

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7240[2].

## Copyright Notice



[1] https://tools.ietf.org/html/rfc5741#section-2
[2] http://www.rfc-editor.org/info/rfc7240
[3] http://trustee.ietf.org/license-info

# 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 in 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 [RFC5023] -- 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 must-understand semantics of the Expect header make it a poor choice for the expression of optional preferences.

Another option available to clients is to utilize Request URI query-string parameters to express preferences. However, any mechanism that alters the URI can have undesirable effects, such as when caches record the altered URI.

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", and "BWS" rules and the #rule extension as defined within Sections 3.2.1 and 3.2.4 of [RFC7230]; as well as the "delta-seconds" rule defined in Section 8.1.3 of [RFC7231].

## 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 are not required for successful completion of the request. Prefer is similar in nature to the Expect header field defined by Section 6.1.2 of [RFC7231] with the exception that servers are allowed to ignore stated preferences.

ABNF:

```
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 5.1). 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 continue processing instead of signaling an error.

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 examples of a "foo" preference with a single "bar" parameter.

```
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. There is no significance given to the ordering of parameters on any given preference.

For both preference token names and parameter names, comparison is case insensitive while values are case sensitive regardless of whether token or quoted-string values are used.

The Prefer header field is end to end and MUST be forwarded by a proxy if the request is forwarded unless Prefer is explicitly identified as being hop by hop using the Connection header field defined by [RFC7230], Section 6.1.

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 "respond-async" preference on their own regardless of whether or not the origin is capable or willing to do so.

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.

A client MAY use multiple instances of the Prefer header field in a single message, or it MAY use a single Prefer header field with multiple comma-separated preference tokens. If multiple Prefer header fields are used, it is equivalent to a single Prefer header field with the comma-separated concatenation of all of the tokens. For example, the following are equivalent:

Multiple Prefer header fields defining three distinct preference tokens:

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

A single Prefer header field defining the same three preference tokens:

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

To avoid any possible ambiguity, individual preference tokens SHOULD NOT appear multiple times within a single request. If any preference is specified more than once, only the first instance is to be considered. All subsequent occurrences SHOULD be ignored without signaling an error or otherwise altering the processing of the request. This is the only case in which the ordering of preferences within a request is considered to be significant.

Due to the inherent complexities involved with properly implementing server-driven content negotiation, effective caching, and the application of optional preferences, implementers are urged to exercise caution when using preferences in a way that impacts the caching of a response and SHOULD NOT use the Prefer header mechanism for content negotiation. If a server supports the optional application of a preference that might result in a variance to a cache's handling of a response entity, a Vary header field MUST be included in the response listing the Prefer header field regardless of whether the client actually used Prefer in the request. Alternatively, the server MAY include a Vary header with the special value "*" as defined by [RFC7231], Section 8.2.1. Note, however, that use of the "Vary: *" header will make it impossible for a proxy to cache the response.

Note that while Preference tokens are similar in structure to HTTP Expect tokens, the Prefer and Expect header fields serve very distinct purposes and preferences cannot be used as expectations.

## 2.1. 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 request cannot be processed within 10 seconds. An undefined "priority" preference is also specified:

```
POST /some-resource HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: respond-async, wait=10
Prefer: priority=5

{...}
```

2. Use lenient processing:

```
POST /some-resource HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: Lenient

{...}
```

3. Use of an optional, undefined parameter on the return=minimal preference:

```
POST /some-resource HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: return=minimal; foo="some parameter"

{...}
```

# 3. The Preference-Applied Response Header Field

The Preference-Applied response header MAY be included within a response message as an indication as to which Prefer tokens were honored by the server and applied to the processing of a request.

ABNF:

```
Preference-Applied = "Preference-Applied" ":" 1#applied-pref
applied-pref = token [ BWS "=" BWS word ]
```

The syntax of the Preference-Applied header differs from that of the Prefer header in that parameters are not included.

Use of the Preference-Applied header is only necessary when it is not readily and obviously apparent that a server applied a given preference and such ambiguity might have an impact on the client's handling of the response. For instance, when using either the "return=representation" or "return=minimal" preferences, a client application might not be capable of reliably determining if the preference was (or was not) applied simply by examining the payload of the response. In such a case, the Preference-Applied header field can be used.

Request:

```
PATCH /my-document HTTP/1.1
Host: example.org
Content-Type: application/example-patch
Prefer: return=representation

[{"op": "add", "path": "/a", "value": 1}]
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Preference-Applied: return=representation
Content-Location: /my-document

{"a": 1}
```

# 4. 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 preferences (see Section 5.1).

## 4.1. The "respond-async" Preference

The "respond-async" 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 "respond-async" preference by returning a 202 (Accepted) response.

ABNF:

```
respond-async = "respond-async"
```

The key motivation for the "respond-async" 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 "respond-async" preference:

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

{Data}
```

An example asynchronous response using 202 (Accepted):

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

While the 202 (Accepted) response status is defined by [RFC7231], little guidance is given on how and when to use the response code and the process for determining the subsequent final result of the operation is left entirely undefined. Therefore, whether and how any given server supports asynchronous responses is an implementation-specific detail that is considered to be out of the scope of this specification.

## 4.2. The "return=representation" and "return=minimal" Preferences

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.

The "return=minimal" preference, on the other hand, indicates that the client wishes the server to return only 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 (OK) 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 = "return" BWS "=" BWS ("representation" / "minimal")
```

When honoring the "return=representation" preference, the returned representation might not be a representation of the effective request URI when the request is affecting another resource. In such cases, the Content-Location header can be used to identify the URI of the returned representation.

The "return=representation" preference is intended 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.

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:

```
PATCH /item/123 HTTP/1.1
Host: example.org
Content-Type: application/example-patch
Prefer: return=representation

1c1
< ABCDEFGHIJKLMNOPQRSTUVWXYZ
---
> BCDFGHJKLMNPQRSTVWXYZ
```

An example response containing the resource representation:

```
HTTP/1.1 200 OK
Content-Location: http://example.org/item/123
Content-Type: text/plain
ETag: "d3b07384d113edec49eaa6238ad5ff00"

BCDFGHJKLMNPQRSTVWXYZ
```

In contrast, the "return=minimal" preference can reduce 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
```

The "return=minimal" and "return=representation" preferences are mutually exclusive directives. It is anticipated that there will never be a situation where it will make sense for a single request to include both

preferences. Any such requests will likely be the result of a coding error within the client. As such, a request containing both preferences can be treated as though neither were specified.

## 4.3. The "wait" Preference

The "wait" preference can be used to establish an upper bound on the length of time, in seconds, the client expects it will take the server to process the request once it has been received. In the case that generating a response will take longer than the time specified, the server, or proxy, can choose to utilize an asynchronous processing model by returning -- for example -- a 202 (Accepted) response.

ABNF:

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

It is important to consider that HTTP messages spend some time traversing the network and being processed by intermediaries. This increases the length of time that a client will wait for a response in addition to the time the server takes to process the request. A client that has strict timing requirements can estimate these factors and adjust the wait value accordingly.

As with other preferences, the "wait" preference could be ignored. Clients can abandon requests that take longer than they are prepared to wait.

For example, a server receiving the following request might choose to respond asynchronously if processing the request will take longer than 10 seconds:

```
POST /collection HTTP/1.1
Host: example.org
Content-Type: text/plain
Prefer: respond-async, wait=10

{Data}
```

## 4.4. The "handling=strict" and "handling=lenient" Processing Preferences

The "handling=strict" and "handling=lenient" preferences indicate, 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 "handling=strict" preference can be used to indicate that, while any particular error may be recoverable, the client would prefer that the server reject the request. The "handling=lenient" preference, on the other hand, indicates that the client wishes the server to attempt to process the request.

ABNF:

```
handling = "handling" BWS "=" BWS ("strict" / "lenient")
```

An example request specifying the "strict" preference:

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

The "handling=strict" and "handling=lenient" preferences are mutually exclusive directives. It is anticipated that there will never be a situation where it will make sense for a single request to include both preferences. Any such requests will likely be the result of a coding error within the client. As such, a request containing both preferences can be treated as though neither were specified.

# 5. IANA Considerations

The 'Prefer' and 'Preference-Applied' header fields have been added to the "Permanent Message Header Field Names" registry defined in [RFC3864] (http://www.iana.org/assignments/message-headers).

Header field name: Prefer

Applicable Protocol: HTTP

Status: Standard

Author: James M Snell <jasnell@gmail.com>

Change controller: IETF

Specification document: this specification, Section 2

Header field name: Preference-Applied

Applicable Protocol: HTTP

Status: Standard

Author: James M Snell <jasnell@gmail.com>

Change controller: IETF

Specification document: this specification, Section 3

## 5.1. The Registry of Preferences

IANA has created a new registry, "HTTP Preferences", under the "Hypertext Transfer Protocol (HTTP) Parameters" registry. New registrations will use the Specification Required policy [RFC5226]. The requirements for registered preferences are described in Section 4.

Registration requests consist of the completed registration template below, typically published in the required specification. However, to allow for the allocation of values prior to publication, the Designated Expert can approve registration based on a separately submitted template once they are satisfied that a specification will be published. 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:

- Preference: (A value for the Prefer request header field that conforms to the syntax rule given in Section 2)
- Value: (An enumeration or description of possible values for the preference token).
- Optional Parameters: (An enumeration of optional parameters, and their values, associated with the preference token).
- Description:
- Reference:
- Notes: [optional]

The "Value" and "Optional Parameters" fields MAY be omitted from the registration template if the specific preference token definition does not define either.

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.

The Expert Reviewer shall ensure:

- That the requested preference name conforms to the token rule in Section 2 and that it is not identical to any other registered preference name;
- That any associated value, parameter names, and values conform to the relevant ABNF grammar specifications in Section 2;

- That the name is 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 uses.
- That requested preferences do not constrain servers, clients, or any intermediaries to any behavior required for successful processing; and
- That the specification document defining the preference includes a proper and complete discussion of any security considerations relevant to the use of the preference.

## 5.2. Initial Registry Contents

The "HTTP Preferences" registry's initial contents are:

- Preference: respond-async
- Description: Indicates that the client prefers that the server respond asynchronously to a request.
- Reference: [this specification], Section 4.1
- Preference: return
- Value: One of either "minimal" or "representation"
- Description: When the value is "minimal", it indicates that the client prefers that the server return a minimal response to a request. When the value is "representation", it indicates that the client prefers that the server include a representation of the current state of the resource in response to a request.
- Reference: [this specification], Section 4.2
- Preference: wait
- Description: Indicates an upper bound to the length of time the client expects it will take for the server to process the request once it has been received.
- Reference: [this specification], Section 4.3
- Preference: handling
- Value: One of either "strict" or "lenient"
- Description: When value is "strict", it indicates that the client wishes the server to apply strict validation and error handling to the processing of a request. When the value is "lenient", it indicates that the client wishes the server to apply lenient validation and error handling to the processing of the request.
- Reference: [this specification], Section 4.4

## 6. Security Considerations

Specific preferences requested by a client can introduce security considerations and concerns beyond those discussed within HTTP/1.1 [RFC7230] and its associated specification documents (see [RFC7230] for the list of associated works). Implementers need to 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.

# 7. References

## 7.1. Normative References

[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.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

## 7.2. Informative References

[RFC5023] Gregorio, J. and B. de hOra, "The Atom Publishing Protocol", RFC 5023, October 2007.

## Author's Address

**James M Snell**
Email: jasnell@gmail.com