Network Working Group
Request for Comments: 2145
Category: Informational

J. C. Mogul
DEC
R. Fielding
UC Irvine
J. Gettys
DEC
H. Frystyk
MIT/LCS
May 1997

Use and Interpretation of HTTP Version Numbers

Status of this Memo

Abstract

HTTP request and response messages include an HTTP protocol version number. Some confusion exists concerning the proper use and interpretation of HTTP version numbers, and concerning interoperability of HTTP implementations of different protocol versions. This document is an attempt to clarify the situation. It is not a modification of the intended meaning of the existing HTTP/1.0 and HTTP/1.1 documents, but it does describe the intention of the authors of those documents, and can be considered definitive when there is any ambiguity in those documents concerning HTTP version numbers, for all versions of HTTP.

TABLE OF CONTENTS

1 Introduction

HTTP request and response messages include an HTTP protocol version number. According to section 3.1 of the HTTP/1.1 specification [2],

> HTTP uses a "<major>.<minor>" numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of a message and its capacity for understanding further HTTP communication, rather than the features obtained via that communication. No change is made to the version number for the addition of message components which do not affect communication behavior or which only add to extensible field values. The <minor> number is incremented when the changes made to the protocol add features which do not change the general message parsing algorithm, but which may add to the message semantics and imply additional capabilities of the sender. The <major> number is incremented when the format of a message within the protocol is changed.

The same language appears in the description of HTTP/1.0 [1].

Many readers of these documents have expressed some confusion about the intended meaning of this policy. Also, some people who wrote HTTP implementations before RFC1945 [1] was issued were not aware of the intentions behind the introduction of version numbers in HTTP/1.0. This has led to debate and inconsistency regarding the use and interpretation of HTTP version numbers, and has led to interoperability problems in certain cases.

This document is an attempt to clarify the situation. It is not a modification of the intended meaning of the existing HTTP/1.0 and HTTP/1.1 documents, but it does describe the intention of the authors of those documents. In any case where either of those two documents is ambiguous regarding the use and interpretation of HTTP version numbers, this document should be considered the definitive as to the intentions of the designers of HTTP.

The specification described in this document is not part of the specification of any individual version of HTTP, such as HTTP/1.0 or HTTP/1.1. Rather, this document describes the use of HTTP version numbers in any version of HTTP (except for HTTP/0.9, which did not include version numbers).

No vendor or other provider of an HTTP implementation should claim any compliance with any IETF HTTP specification unless the implementation conditionally complies with the rules in this document.

1.1 Robustness Principle

RFC791 [4] defines the "robustness principle" in section 3.2:

> an implementation must be conservative in its sending behavior, and liberal in its receiving behavior.

This principle applies to HTTP, as well. It is the fundamental basis for interpreting any part of the HTTP specification that might still be ambiguous. In particular, implementations of HTTP SHOULD NOT reject messages or generate errors unnecessarily.

2 HTTP version numbers

We start by restating the language quoted above from section 3.1 of the HTTP/1.1 specification [2]:

> It is, and has always been, the explicit intent of the HTTP specification that the interpretation of an HTTP message header does not change between minor versions of the same major version.
>
> It is, and has always been, the explicit intent of the HTTP specification that an implementation receiving a message header that it does not understand MUST ignore that header. (The word "ignore" has a special meaning for proxies; see section 2.1 below.)

To make this as clear as possible: The major version sent in a message MAY indicate the interpretation of other header fields. The minor version sent in a message MUST NOT indicate the interpretation of other header fields. This reflects the principle that the minor version labels the capability of the sender, not the interpretation of the message.

> Note: In a future version of HTTP, we may introduce a mechanism that explicitly requires a receiving implementation to reject a message if it does not understand certain headers. For example, this might be implemented by means of a header that lists a set of other message headers that must be understood by the recipient. Any implementation claiming at least conditional compliance with this future version of HTTP would have to implement this mechanism. However, no implementation claiming compliance with a lower HTTP version (in particular, HTTP/1.1) will have to implement this mechanism.
>
> This future change may be required to support the Protocol Extension Protocol (PEP) [3].

One consequence of these rules is that an HTTP/1.1 message sent to an HTTP/1.0 recipient (or a recipient whose version is unknown) MUST be constructed so that it remains a valid HTTP/1.0 message when all headers not defined in the HTTP/1.0 specification [1] are removed.

2.1 Proxy behavior

A proxy MUST forward an unknown header, unless it is protected by a Connection header. A proxy implementing an HTTP version >= 1.1 MUST NOT forward unknown headers that are protected by a Connection header, as described in section 14.10 of the HTTP/1.1 specification [2].

We remind the reader that that HTTP version numbers are hop-by-hop components of HTTP messages, and are not end-to-end. That is, an HTTP proxy never "forwards" an HTTP version number in either a request or response.

2.2 Compatibility between minor versions of the same major version

An implementation of HTTP/x.b sending a message to a recipient whose version is known to be HTTP/x.a, a < b, MAY send a header that is not defined in the specification for HTTP/x.a. For example, an HTTP/1.1 server may send a "Cache-control" header to an HTTP/1.0 client; this may be useful if the immediate recipient is an HTTP/1.0 proxy, but the ultimate recipient is an HTTP/1.1 client.

An implementation of HTTP/x.b sending a message to a recipient whose version is known to be HTTP/x.a, a < b, MUST NOT depend on the recipient understanding a header not defined in the specification for HTTP/x.a. For example, HTTP/1.0 clients cannot be expected to understand chunked encodings, and so an HTTP/1.1 server must never send "Transfer-Encoding: chunked" in response to an HTTP/1.0 request.

2.3 Which version number to send in a message

The most strenuous debate over the use of HTTP version numbers has centered on the problem of implementations that do not follow the robustness principle, and which fail to produce useful results when they receive a message with an HTTP minor version higher than the minor version they implement. We consider these implementations buggy, but we recognize that the robustness principle also implies that message senders should make concessions to buggy implementations when this is truly necessary for interoperation.

An HTTP client SHOULD send a request version equal to the highest version for which the client is at least conditionally compliant, and whose major version is no higher than the highest version supported by the server, if this is known. An HTTP client MUST NOT send a version for which it is not at least conditionally compliant.

An HTTP client MAY send a lower request version, if it is known that the server incorrectly implements the HTTP specification, but only after the client has determined that the server is actually buggy.

An HTTP server SHOULD send a response version equal to the highest version for which the server is at least conditionally compliant, and whose major version is less than or equal to the one received in the request. An HTTP server MUST NOT send a version for which it is not at least conditionally compliant. A server MAY send a 505 (HTTP Version Not Supported) response if cannot send a response using the major version used in the client’s request.

An HTTP server MAY send a lower response version, if it is known or suspected that the client incorrectly implements the HTTP specification, but this should not be the default, and this SHOULD NOT be done if the request version is HTTP/1.1 or greater.

3 Security Considerations

None, except to the extent that security mechanisms introduced in one version of HTTP might depend on the proper interpretation of HTTP version numbers in older implementations.

4 References

1. Berners-Lee, T., R. Fielding, and H. Frystyk. Hypertext Transfer Protocol -- HTTP/1.0. RFC 1945, HTTP Working Group, May, 1996.

2. Fielding, Roy T., Jim Gettys, Jeffrey C. Mogul, Henrik Frystyk Nielsen, and Tim Berners-Lee. Hypertext Transfer Protocol -- HTTP/1.1. RFC 2068, HTTP Working Group, January, 1997.

3. Khare, Rohit. HTTP/1.2 Extension Protocol (PEP). HTTP Working Group, Work in Progress.

4. Postel, Jon. Internet Protocol. RFC 791, NIC, September, 1981.

5 Authors' addresses

Jeffrey C. Mogul
Western Research Laboratory
Digital Equipment Corporation
250 University Avenue
Palo Alto, California, 94305, USA
Email: mogul@wrl.dec.com

Roy T. Fielding
Department of Information and Computer Science
University of California
Irvine, CA 92717-3425, USA
Fax: +1 (714) 824-4056
Email: fielding@ics.uci.edu

Jim Gettys
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA
Fax: +1 (617) 258 8682
Email: jg@w3.org

Henrik Frystyk Nielsen
W3 Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, USA
Fax: +1 (617) 258 8682
Email: frystyk@w3.org