Internet Engineering Task Force (IETF)
Request for Comments: 6585
Updates: 2616
Category: Standards Track
ISSN: 2070-1721

M. Nottingham
Rackspace
R. Fielding
Adobe

April 2012

Additional HTTP Status Codes

Abstract

This document specifies additional HyperText Transfer Protocol (HTTP) status codes for a variety of common situations.

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.

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/rfc6585.

Table of Contents

1. Introduction

This document specifies additional HTTP [RFC2616] status codes for a variety of common situations, to improve interoperability and avoid confusion when other, less precise status codes are used.

Note that these status codes are optional; servers cannot be required to support them. However, because clients will treat unknown status codes as a generic error of the same class (e.g., 499 is treated as 400 if it is not recognized), they can be safely deployed by existing servers (see [RFC2616] Section 6.1.1 for more information).

2. Requirements

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

3. 428 Precondition Required

The 428 status code indicates that the origin server requires the request to be conditional.

Its typical use is to avoid the "lost update" problem, where a client GETs a resource’s state, modifies it, and PUTs it back to the server, when meanwhile a third party has modified the state on the server, leading to a conflict. By requiring requests to be conditional, the server can assure that clients are working with the correct copies.

Responses using this status code SHOULD explain how to resubmit the request successfully. For example:

```
HTTP/1.1 428 Precondition Required
Content-Type: text/html
```

```
<html>
   <head>
      <title>Precondition Required</title>
   </head>
   <body>
      <h1>Precondition Required</h1>
      <p>This request is required to be conditional;
      try using "If-Match".</p>
   </body>
</html>
```

Responses with the 428 status code MUST NOT be stored by a cache.

4. 429 Too Many Requests

The 429 status code indicates that the user has sent too many requests in a given amount of time ("rate limiting").

The response representations SHOULD include details explaining the condition, and MAY include a Retry-After header indicating how long to wait before making a new request.

For example:

```
HTTP/1.1 429 Too Many Requests
Content-Type: text/html
Retry-After: 3600

<html>
   <head>
      <title>Too Many Requests</title>
   </head>
   <body>
      <h1>Too Many Requests</h1>
      <p>I only allow 50 requests per hour to this Web site per
         logged in user.  Try again soon.</p>
   </body>
</html>
```

Note that this specification does not define how the origin server identifies the user, nor how it counts requests. For example, an origin server that is limiting request rates can do so based upon counts of requests on a per-resource basis, across the entire server, or even among a set of servers. Likewise, it might identify the user by its authentication credentials, or a stateful cookie.

Responses with the 429 status code MUST NOT be stored by a cache.

5. 431 Request Header Fields Too Large

The 431 status code indicates that the server is unwilling to process the request because its header fields are too large. The request MAY be resubmitted after reducing the size of the request header fields.

It can be used both when the set of request header fields in total is too large, and when a single header field is at fault. In the latter case, the response representation SHOULD specify which header field was too large.

For example:

```
HTTP/1.1 431 Request Header Fields Too Large
Content-Type: text/html

<html>
   <head>
      <title>Request Header Fields Too Large</title>
   </head>
   <body>
      <h1>Request Header Fields Too Large</h1>
      <p>The "Example" header was too large.</p>
   </body>
</html>
```

Responses with the 431 status code MUST NOT be stored by a cache.

6. 511 Network Authentication Required

The 511 status code indicates that the client needs to authenticate to gain network access.

The response representation SHOULD contain a link to a resource that allows the user to submit credentials (e.g., with an HTML form).

Note that the 511 response SHOULD NOT contain a challenge or the login interface itself, because browsers would show the login interface as being associated with the originally requested URL, which may cause confusion.

The 511 status SHOULD NOT be generated by origin servers; it is intended for use by intercepting proxies that are interposed as a means of controlling access to the network.

Responses with the 511 status code MUST NOT be stored by a cache.

6.1. The 511 Status Code and Captive Portals

The 511 status code is designed to mitigate problems caused by "captive portals" to software (especially non-browser agents) that is expecting a response from the server that a request was made to, not the intervening network infrastructure. It is not intended to encourage deployment of captive portals -- only to limit the damage caused by them.

A network operator wishing to require some authentication, acceptance of terms, or other user interaction before granting access usually does so by identifying clients who have not done so ("unknown clients") using their Media Access Control (MAC) addresses.

Unknown clients then have all traffic blocked, except for that on TCP port 80, which is sent to an HTTP server (the "login server") dedicated to "logging in" unknown clients, and of course traffic to the login server itself.

For example, a user agent might connect to a network and make the following HTTP request on TCP port 80:

```
GET /index.htm HTTP/1.1
Host: www.example.com
```

Upon receiving such a request, the login server would generate a 511 response:

```
HTTP/1.1 511 Network Authentication Required
Content-Type: text/html

<html>
   <head>
      <title>Network Authentication Required</title>
      <meta http-equiv="refresh"
            content="0; url=https://login.example.net/">
   </head>
   <body>
      <p>You need to <a href="https://login.example.net/">
      authenticate with the local network</a> in order to gain
      access.</p>
   </body>
</html>
```

Here, the 511 status code assures that non-browser clients will not interpret the response as being from the origin server, and the META HTML element redirects the user agent to the login server.

7. Security Considerations

7.1. 428 Precondition Required

The 428 status code is optional; clients cannot rely upon its use to prevent "lost update" conflicts.

7.2. 429 Too Many Requests

When a server is under attack or just receiving a very large number of requests from a single party, responding to each with a 429 status code will consume resources.

Therefore, servers are not required to use the 429 status code; when limiting resource usage, it may be more appropriate to just drop connections, or take other steps.

7.3. 431 Request Header Fields Too Large

Servers are not required to use the 431 status code; when under attack, it may be more appropriate to just drop connections, or take other steps.

7.4. 511 Network Authentication Required

In common use, a response carrying the 511 status code will not come from the origin server indicated in the request’s URL. This presents many security issues; e.g., an attacking intermediary may be inserting cookies into the original domain’s name space, may be observing cookies or HTTP authentication credentials sent from the user agent, and so on.

However, these risks are not unique to the 511 status code; in other words, a captive portal that is not using this status code introduces the same issues.

Also, note that captive portals using this status code on a Secure Socket Layer (SSL) or Transport Layer Security (TLS) connection (commonly, port 443) will generate a certificate error on the client.

8. IANA Considerations

The HTTP Status Codes registry has been updated with the following entries:

Value: 428
Description: Precondition Required
Reference: [RFC6585]

Value: 429
Description: Too Many Requests
Reference: [RFC6585]

Value: 431
Description: Request Header Fields Too Large
Reference: [RFC6585]

Value: 511
Description: Network Authentication Required
Reference: [RFC6585]

9. References

9.1. Normative References

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

9.2. Informative References

[CORS] van Kesteren, A., Ed., "Cross-Origin Resource Sharing", W3C Working Draft WD-cors-20100727, July 2010, <http://www.w3.org/TR/cors/>.

[Favicon] Wikipedia, "Favicon", March 2012, <http://en.wikipedia.org/w/index.php?title=Favicon&oldid=484627550>.

[OAuth2.0] Hammer-Lahav, E., Ed., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Protocol", Work in Progress, March 2012.

[P3P] Marchiori, M., Ed., "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification", W3C Recommendation REC-P3P-20020416, April 2002, <http://www.w3.org/TR/P3P/>.

[RFC4791] Daboo, C., Desruisseaux, B., and L. Dusseault, "Calendaring Extensions to WebDAV (CalDAV)", RFC 4791, March 2007.

[RFC4918] Dusseault, L., Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007.

[WIDGETS] Caceres, M., Ed., "Widget Packaging and XML Configuration", W3C Recommendation REC-widgets-20110927, September 2011, <http://www.w3.org/TR/widgets/>.

[WebFinger] WebFinger Project, "WebFingerProtocol (Draft)", January 2010, <http://code.google.com/p/webfinger/wiki/WebFingerProtocol>.

Appendix A. Acknowledgements

Thanks to Jan Algermissen and Julian Reschke for their suggestions and feedback.

Appendix B. Issues Raised by Captive Portals

Since clients cannot differentiate between a portal's response and that of the HTTP server that they intended to communicate with, a number of issues arise. The 511 status code is intended to help mitigate some of them.

One example is the "favicon.ico" [Favicon] commonly used by browsers to identify the site being accessed. If the favicon for a given site is fetched from a captive portal instead of the intended site (e.g., because the user is unauthenticated), it will often "stick" in the browser's cache (most implementations cache favicons aggressively) beyond the portal session, so that it seems as if the portal's favicon has "taken over" the legitimate site.

Another browser-based issue comes about when the Platform for Privacy Preferences [P3P] is supported. Depending on how it is implemented, it's possible a browser might interpret a portal's response for the p3p.xml file as the server's, resulting in the privacy policy (or lack thereof) advertised by the portal being interpreted as applying to the intended site. Other Web-based protocols such as WebFinger [WebFinger], Cross-Origin Resource Sharing [CORS], and Open Authorization [OAuth2.0] may also be vulnerable to such issues.

Although HTTP is most widely used with Web browsers, a growing number of non-browsing applications use it as a substrate protocol. For example, Web Distributed Authoring and Versioning (WebDAV) [RFC4918] and Calendaring Extensions to WebDAV (CalDAV) [RFC4791] both use HTTP as the basis (for remote authoring and calendaring, respectively). Using these applications from behind a captive portal can result in spurious errors being presented to the user, and might result in content corruption, in extreme cases.

Similarly, other non-browser applications using HTTP can be affected as well, e.g., widgets [WIDGETS], software updates, and other specialized software such as Twitter clients and the iTunes Music Store.

It should be noted that it's sometimes believed that using HTTP redirection to direct traffic to the portal addresses these issues. However, since many of these uses "follow" redirects, this is not a good solution.

Authors' Addresses

Mark Nottingham
Rackspace

EMail: mnot@mnot.net
URI: http://www.mnot.net/

Roy T. Fielding
Adobe Systems Incorporated
345 Park Ave.
San Jose, CA 95110
USA

EMail: fielding@gbiv.com
URI: http://roy.gbiv.com/