RFC 6585 : Additional HTTP Status Codes

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 mai 2012

Date de publication du RFC : Avril 2012

https://www.bortzmeyer.org/6585.html

Le protocole HTTP, normalisé dans le RFC 7230 [1], utilise des réponses numériques sur trois chiffres pour indiquer le résultat de la requête. Certains de ces codes sont devenus célèbres comme le 404 pour une ressource non trouvée. Et comme l'Internet fonctionne avec des LOLcats, il existe même un ensemble de photos des réponses HTTP avec des chats <http://www.flickr.com/photos/girliemac/sets/72157628409467125/with/6508023065/> (d'où sont tirées les images de cet article). Mais la liste des réponses possibles n'est pas figée et de nouveaux codes sont ajoutés de temps en temps. Ce RFC normalise quatre de ces codes.

Ces nouveaux codes permettent de préciser le code utilisé autrefois, qui était en général moins précis. Ainsi, les limiteurs de trafic utilisaient souvent, lorsqu'un client demandait trop de ressources, le code 403 (Interdiction), qui pouvait servir à bien d'autres choses. Ils ont désormais un code à eux, 429. Les nouveaux codes sont enregistrés à l'IANA <https://www.iana.org/assignments/http-status-codes/http-status-codes.xml>.

Rappelez-vous bien que le premier chiffre indique la classe de l'erreur (RFC 7231, section 6) et qu'un serveur HTTP qui utilise un code nouveau n'a pas de souci à se faire : un client qui ne connaîtrait pas ce nouveau code peut le traiter comme le code générique commençant par le même chiffre. Pour reprendre l'excellent résumé de Dana Contreras, « *"HTTP response codes for dummies. 5xx : we fucked up. 4xx : you fucked up. 3xx : ask that dude over there. 2xx : cool"* ».

Premier cité (section 3), 428 **Precondition Required** (pas de jolie image de chat pour ce code). Pour éviter le problème de la « mise à jour perdue » (un client récupère une ressource Web, la modifie et la renvoie au serveur pour mise à jour, alors que, pendant ce temps, un autre client a fait pareil : l'un des deux va perdre sa mise à jour), certains serveurs HTTP exigent que les requêtes soient conditionnelles (par exemple avec le `If-Match:` de la section 3.1 du RFC 7232). Si le client n'a pas utilisé de condition, le serveur l'enverra promener avec ce code 428 (qui doit normalement être accompagné d'un message clair indiquant le problème) :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7230.txt

```
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>
```

Deuxième code, 429 **Too Many Requests** (section 4) Il était apparemment assez répandu dans la nature mais n'était pas encore normalisé. C'est désormais fait et les limiteurs de trafic peuvent donc désormais dire clairement à leurs clients qu'ils exagèrent :

```
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>
```

C'est uniquement un indicateur d'une décision prise par le serveur : le RFC n'impose pas aux serveurs un mécanisme particulier pour prendre cette décision. De même, le serveur n'est pas **obligé** de renvoyer quelque chose (ce qui consomme des ressources), il a le droit d'ignorer purement et simplement les requêtes excessives (section 7.2).

Troisième code, en section 5, 431 **Request Header Fields Too Large** Le protocole HTTP n'impose pas de limite à la taille des en-têtes envoyés lors d'une requête mais un serveur donné peut avoir une telle limite. Si elle est dépassée, il peut répondre 431. Le même code peut dire « taille totale des en-têtes trop grande » ou « taille d'un en-tête trop grande ». Dans ce dernier cas, la réponse doit indiquer quel en-tête :

```
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>
```

Ces trois premiers codes n'ont pas suscité de controverses particulières. Ce n'était pas le cas de 511 **Network Authentication Required** (section 6, pas de jolie image de chat pour ce code) qui indique que le client aurait dû s'authentifier. Le problème est d'architecture : ce code est utilisé pour les portails captifs qui, tant qu'on n'est pas authentifié, redirigent d'autorité toutes les requêtes HTTP vers un serveur permettant de s'authentifier. (Après l'authentification, l'adresse MAC est mémorisée par le routeur, qui ne redirige plus.) De tels portails violent sérieusement les bons principes d'architecture du Web. Mais ils existent et il était donc préférable qu'ils puissent répondre avec un code sans ambiguité. La réponse 511 inclut donc le lien vers la page de "*login*". Elle ne doit jamais être envoyée par un serveur normal, uniquement par le relais détourneur.

Le RFC dit donc que les portails captifs sont le Mal (surtout pour les clients HTTP qui ne sont pas des navigateurs) mais que le 511 permettra de limiter les dégats. Au moins, curl ou wget comprendront tout de suite ce qui s'est passé.

Un exemple de réponse (notez l'élément `<meta>` pour les navigateurs) :

```
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>
```

Les réponses 511 ou, plus exactement, les redirections forcées posent des tas de problèmes de sécurité. Par exemple, elles ne viennent pas du serveur demandé, ce qui casse certains mécanismes de sécurité de HTTP comme les "*cookies*" du RFC 6265. Un problème analogue survient si on utilise TLS car le certificat X.509 ne correspondra pas au nom demandé. L'annexe B du RFC détaille les problèmes de sécurité des portails captifs et les façons de les limiter. Cela va de problèmes esthétiques (la "*favicon*" récupérée depuis le portail et qui s'accroche ensuite dans le cache du navigateur) jusqu'aux protocoles de sécurité qui risquent de prendre la réponse du portail pour l'information qu'ils demandaient (P3P, WebFinger, OAuth, etc). Ces protocoles sont souvent utilisés par des applications HTTP qui ne passent pas par un navigateur (Twitter, iTunes, le WebDAV du RFC 4918, etc). En détectant le code 511, ces applications ont une chance de pouvoir réagir proprement (en ignorant les réponses transmises avec ce code).

À l'heure actuelle, je ne pense pas qu'il y ait beaucoup de "*hotspots*" qui utilisent ce code mais cela viendra peut-être. Mais je n'en suis pas sûr, ces portails captifs étant en général programmés avec les pieds, par un auteur anonyme à qui on ne peut jamais signaler les bogues. Et je ne sais pas quels sont les clients HTTP qui le reconnaissent et agissent intelligement. Si un lecteur courageux veut enquêter...

Les images de chats citées plus haut sont également accessibles via une API REST simple `<http://httpcats.herokuapp.com/>`. Si vous préférez les chiens, ça existe aussi `<http://httpstatusdogs.com/>`.