RFC 8942 : HTTP Client Hints

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 9 février 2021

Date de publication du RFC : Février 2021

https://www.bortzmeyer.org/8942.html

Aux débuts du Web, il allait de soi que le contenu renvoyé à un client était identique quel que soit le client. On demandait un article scientifique, le même article était donné à tout le monde. La séparation du contenu et de la présentation que permet HTML faisait en sorte que ce contenu s'adaptait automatiquement aux différents clients et notamment à leur taille. Mais petit à petit l'habitude s'est prise d'envoyer un contenu différent selon le client. C'était parfois pour de bonnes raisons (s'adapter à la langue de l'utilisateur) et parfois pour des mauvaises (incompréhension de la séparation du contenu et de la présentation, ignorance du Web par des commerciaux qui voulaient contrôler l'apparence exacte de la page). Cette adaptation au client peut se faire en tenant compte des **en-têtes** envoyés par le navigateur dans sa requête, notamment l'affreux `User-Agent:`, très indiscret et en général mensonger. Mais la tendance actuelle est de ne pas utiliser systématiquement ces en-têtes, très dangereux pour la vie privée et parfois inutiles : si le serveur HTTP n'adapte pas le contenu en fonction des en-têtes, pourquoi les envoyer ? Ce nouveau RFC propose une solution : un en-tête `Accept-CH:` envoyé par le **serveur** qui indique ce que le serveur va faire des en-têtes d'indication envoyés par le client ("*client hints*") dans ses futures requêtes. Il s'agit d'un projet Google, déjà mis en œuvre dans Chrome mais, pour l'instant, seulement avec le statut « Expérimental ».

Le Web a toujours été prévu pour être accédé par des machines très différentes, en terme de taille d'écran, de logiciels utilisés, de capacités de traitement. Sans compter les préférences propres à l'utilisateur, par exemple sa langue. C'est ainsi que, par exemple, les lignes de texte ne sont pas définies dans le source en HTML, elles seront calculées dynamiquement par le navigateur qui, lui, connait la largeur de la fenêtre. Mais il a été difficile de faire comprendre cela à des marketeux habitués de la page imprimée et qui insistaient pour contrôler tout au pixel près.

Bref, il y a longtemps que des gens qui ne connaissent pas le Web font n'importe quoi, par exemple en regardant l'en-tête `User-Agent:` (RFC 7231 [1], section 5.5.3) et en ajustant le contenu Web à cet en-tête, ce qui nécessite une base de données de tous les `User-Agent:` possibles puisque chaque nouvelle version d'un navigateur peut changer les choses. (D'où ces `User-Agent:` ridicules et mensongers

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc7231.txt`

qu'on voit aujourd'hui, où chaque navigateur met le nom de tous les autres, par exemple Edge annonce `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.2704.79 Safari/537.36 Edge/18.014`.) Ces techniques sont évidemment très mauvaises : elles compliquent les serveurs, elles ne permettent pas de gérer le cas d'un nouveau navigateur (sauf à ce qu'il annonce les noms de ces prédécesseurs dans `User-Agent:`, ce qui se fait couramment), elles ne permettent pas facilement à l'utilisateur ou à son navigateur de savoir quels critères ont été utilisés par le serveur pour adapter le contenu. Et elles sont très indiscrètes, exposant par défaut bien plus d'informations que ce qui est nécessaire au serveur pour adapter son contenu. C'est ainsi que des services comme Panopticlick <https://panopticlick.eff.org/> ou "*Am I unique?*" <https://amiunique.org/> peuvent fonctionner, démontrant la possibilité d'identifier un utilisateur sans biscuits, avec juste du "*fingerprinting*" passif.

Un site Web peut aussi ajuster son contenu en fonction d'un utilisateur particulier, suivi, par exemple, par les biscuits du RFC 6265, ou via des paramètres dans l'URL mais cela complique les choses, par exemple en obligeant à mettre des informations dans l'URL de chaque ressource du site Web.

Une meilleure solution, au moins en théorie, est la négociation de contenu HTTP (RFC 7231, section 3.4.1) : le client utilise alors des en-têtes bien définis (comme `Accept-Language:` pour la langue) auxquels le serveur peut répondre par un contenu spécifique. La négociation de contenu <https://www.bortzmeyer.org/negociation-contenu-http.html> pose toutefois un problème, note le RFC, c'est que les en-têtes sont systématiquement envoyés par le client, qui ne sait pas si le serveur en fera quelque chose. Cela peut notamment avoir des conséquences en terme de vie privée, un client pouvant être identifié par le jeu complet des en-têtes qu'il envoie, même en l'absence de biscuits de traçage. (En outre, pour le cas particulier de la langue, la négociation de contenu n'aide guère car elle ne prend pas en compte des questions comme la notion de version originale <https://www.bortzmeyer.org/web-et-version-originale.html>.)

Au contraire, estime le RFC (avec pas mal d'optimisme), le mécanisme décrit ici peut potentiellement être davantage protecteur, en obligeant le serveur à annoncer quelles informations il va utiliser et en n'envoyant les informations permettant l'ajustement du contenu que lorsqu'elles sont explicitement demandées.

En quoi consiste ce mécanisme, justement ? La section 2 du RFC le décrit. Un en-tête CH ("*Client Hint*", indication par le client) est un en-tête HTTP qui va envoyer des données permettant au serveur de modifier le contenu. Les clients HTTP (les navigateurs Web, par exemple) envoient les en-tête CH en fonction de leur propre configuration, et des demandes du serveur, via l'en-tête `Accept-CH:`. Le projet de spécification « "*Client Hints Infrastructure*" <https://wicg.github.io/client-hints-infrastructure/> » discute de telles politiques et donne un jeu minimal d'indications qui n'identifient pas trop le client. Notre RFC ajoute que, par défaut, le client HTTP ne devrait pas envoyer de données autres que celles contenues dans ce jeu minimal. On verra bien si les navigateurs Web suivront cette recommandation. En tout cas, le RFC insiste bien sur le risque d'identification ("*fingerprinting*"), avec toute technique où le client exprime des préférences. Notez qu'à l'heure actuelle, les indications client ne sont pas encore spécifiées, mais il existe des propositions <https://wicg.github.io/ua-client-hints/>.

Si le client envoie des en-têtes CH ("*Client Hint*"), le serveur peut alors ajuster son contenu. Comme celui-ci dépend des en-têtes CH, le serveur doit penser à ajouter un en-tête `Vary:` (RFC 7231, section 7.1.4). Le serveur doit ignorer les indications qu'il ne comprend pas (ce qui permettra d'en déployer des nouvelles sans tout casser).

Comment est-ce que le serveur HTTP indique qu'il accepte le système des indications client ("*client hints*") ? La section 3 de notre RFC décrit l'en-tête `Accept-CH:` ("*Accept Client Hints*") qui sert à indiquer que le serveur connait bien ce système. Il figure désormais dans le registre des en-têtes <https://www.iana.org/assignments/message-headers/message-headers.xml#perm-headers>. `Accept-CH:` est un en-tête structuré (RFC 8941) et sa valeur est une liste des indications qu'il accepte. Ainsi :

```
Accept-CH: CH-Example, CH-Example-again
```

indique que le serveur HTTP connait le système décrit dans notre RFC et qu'il utilise deux indications client, `CH-Example` et `CH-Example-again`. (À l'heure actuelle, il n'y a pas encore d'indications client normalisées, donc le RFC et cet article utilisent des exemples bidons.)

Voici un exemple plus complet. Le serveur veut savoir quel logiciel utilise le client et sur quel système d'exploitation. Il envoie :

```
Accept-CH: Sec-CH-UA-Full-Version, Sec-CH-UA-Platform
```

(UA = "*User Agent*", le client HTTP, il s'agit ici de propositions d'indications client `<https://wicg.github.io/ua-client-hints/>`; `Sec-` et `CH-` sont expliqués plus loin.) Le navigateur va alors, dans ses requêtes suivantes, et s'il est d'accord, envoyer :

```
Sec-CH-UA: "Examplary Browser"; v="73"
Sec-CH-UA-Full-Version: "73.3R8.2H.1"
Sec-CH-UA-Platform: "Windows"
```

(Notez que cet exemple particulier suppose que `Sec-CH-UA:` est envoyé systématiquement, même non demandé, point qui est encore en discussion.)

Ce système des indications client pose évidemment des problèmes de sécurité, analysés dans la section 4. D'abord, si les indications client ont été conçues pour mieux respecter la vie privée, elles ne sont pas non plus une solution magique. Un serveur malveillant peut indiquer une liste très longue dans son `Accept-CH:` espérant que les clients naïfs lui enverront plein d'informations. (Notons que les chercheurs en sécurité pourront analyser les réponses des serveurs HTTP et compiler des listes de ceux qui abusent, contrairement au système actuel où le serveur peut capter l'information de manière purement passive.) Le client prudent ne doit envoyer au serveur que ce qui est déjà accessible audit serveur par d'autres moyens (API JavaScript par exemple). Et il doit tenir compte des critères suivants :

— Entropie : il ne faut pas envoyer d'information rare, qui pourrait aider à identifier une machine. Indiquer qu'on tourne sur Windows n'est pas bien grave, cela n'est pas rare. Mais un client tournant sur Plan 9 ne devrait pas s'en vanter auprès du serveur.
— Sensibilité : les informations sensibles ne devraient pas être envoyées (le RFC ne donne pas d'exemple mais cela inclut certainement la localisation).
— Stabilité : les informations qui changent rapidement ne devraient pas être envoyées.

À noter que des indications apparemment utiles, comme la langue, peuvent être dangereuses ; s'il s'agit d'une langue rare, elle peut aider à l'identification d'un utilisateur, s'il s'agit de la langue d'une minorité persécutée, elle peut être une information sensible.

Le RFC recommande (reste à savoir si cela sera suivi) que les clients HTTP devraient lier le `Accept-CH:` à une origine (RFC 6454) et n'envoyer donc les informations qu'à l'origine qui les demande. Ils devraient également être configurables, permettant à l'utilisateur d'envoyer plus ou moins d'informations selon son degré de méfiance. (Le RFC n'en parle pas, mais j'ajoute que la valeur par défaut - très bavarde ou au contraire très discrète - est un choix crucial, car peu d'utilisateurs changeront ce réglage.) Et le RFC demande que tout logiciel client permette également la discrétion totale, en n'envoyant aucune indication client. (André Sintzoff me fait remarquer à juste titre que « l'absence d'information est déjà

une information ≫. Signaler qu'on ne veut pas être suivi à la trace peut vous signaler comme élement suspect, à surveiller.)

La liste des indications à envoyer pour chaque origine doit évidemment être remise à zéro lorsqu'on effectue des opérations de nettoyage, comme de supprimer les biscuits.

La même section 4 donne des conseils aux auteurs de futures indications client. On a vu qu'à l'heure actuelle aucune n'était encore normalisée mais des projets existent déjà `<https://wicg.github.io/ua-client-hints/>`, notamment pour remplacer `User-Agent:`. Un des choix cruciaux est de décider si les indications peuvent être générées par une application (typiquement du code JavaScript exécuté par le navigateur Web) ou bien seulement par le client. Les secondes sont évidemment plus sûres et c'est pour cela que leur nom est préfixée par `Sec-` (pour "*Secure*"). C'est une idée qui vient de la spécification Fetch `<https://fetch.spec.whatwg.org/>`.

Toujours côté nommage des indications client, le RFC recommande que le nom soit préfixé de `CH-` pour "*Client Hints*", ce qui peut permettre de les distinguer facilement.

À l'heure actuelle, Chromium met déjà en œuvre ce système des indications client.