

LE PROTOCOLE HTTP.

Patrick Poulingeas.

22 mars 2005.

1. Introduction.

L'abréviation HTTP signifie HyperText Transfer Protocol. Il s'agit d'une norme régissant les échanges (demande d'un document, envoi d'informations, envoi d'un document) entre un client Web (un navigateur par exemple) et un serveur Web. HTTP est un protocole texte (sur 8 bits) qui utilise une liaison TCP pour la communication entre le client et le serveur.

Contrairement à ce qui passe pour les protocoles FTP (transfert de fichiers), SMTP (envoi de mails), POP3 (consultation d'une boîte aux lettres), la liaison entre un client et un serveur Web est généralement brève. On a souvent le scénario suivant :

- 1) Le client établit une connexion TCP avec le serveur Web (habituellement sur le port 80) et émet une demande de document en se servant de l'instruction GET du protocole HTTP.
- 2) Le serveur reçoit la demande, l'examine et en cas de succès retourne le document demandé.
- 3) Le serveur clôture ensuite la session.

On dit qu'HTTP est un protocole sans états (*stateless protocol*). La liaison entre le client et le serveur n'est pas maintenue jusqu'à ce que le client décide d'y mettre fin. Ceci pose un problème pour l'identification d'un client lors d'une session (une série d'achats sur un site de commerce électronique par exemple), le serveur ne pouvant se baser sur la seule adresse IP du client. Divers mécanismes ont donc été mis en place pour la gestion d'une session, dès le niveau HTTP avec les cookies. La version 1.1 de HTTP permet d'utiliser une connexion persistante entre le client et le serveur, mais pour un nombre limité de demandes du client (typiquement pour récupérer des images et d'autres composants multimédia constituant un document Web).

Il existe deux versions majeures du protocole HTTP : la version 1.0 (décrite dans la RFC 1945) et la version 1.1 (décrite dans la RFC 2616).

2. Format d'une requête et d'une réponse HTTP.

2.1. Format des requêtes HTTP envoyées par le client.

Voyons un exemple de requête formulée par un navigateur.

Dans Internet Explorer 6.0, on demande le document référencé par l'URL suivante :

<http://www.msi.unilim.fr/~poulingeas/index.htm>

Le navigateur génère alors une requête HTTP de la forme :

```
GET /~poulingeas/index.htm HTTP/1.1
Accept: image/gif, image/jpeg, */*
Accept-Language: fr
Host: www.msi.unilim.fr
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
```

Une requête HTTP est composée de deux parties principales :

- un en-tête contenant des lignes dont chacune se termine par la suite de caractères <CR><LF> (c'est-à-dire les codes ASCII 13 et 10).
- Un corps **éventuel** qui contient les informations envoyées au serveur dans le cas d'une commande POST.

L'en-tête et le corps éventuel sont séparés par une ligne vide (c'est-à-dire la séquence <CR><LF>).

L'en-tête débute par une ligne de commande de la forme :

Méthode URI VersionHTTP

Exemple :

```
GET /~poulingeas/index.htm HTTP/1.1
```

Les principales méthodes (qui correspondent à des commandes envoyées au serveur) sont :

- GET Demande d'un document au serveur
- HEAD Demande de renseignements sur un document (date de création, taille du document, etc.)
- POST Envoi d'informations au serveur (Par exemple quand on remplit un formulaire dans un site)

Les noms des commandes doivent être en majuscules, sinon elles ne sont pas reconnues par le serveur.

Une URI (*Uniform Ressource Identifier*) est une chaîne de caractères identifiant une ressource physique ou abstraite. Une URL (*Uniform Ressource Locator*) est une URI qui identifie une ressource à travers un mécanisme d'accès (par exemple un protocole réseau + un chemin d'accès).

Exemples d'URL :

```
http://www.library.cornell.edu/nr/cbookcpdf.html
ftp://ftp.lip6.fr/pub/gnat/README
```

Exemples d'URI :

http://www.library.cornell.edu/nr/cbookcpdf.html	URI absolue
/nr/cbookcpdf.html	URI relative
ftp://ftp.lip6.fr/pub/gnat/README	URI absolue
man:ls(1)	Page de manuel de la commande ls

Enfin, la version du protocole HTTP utilisée par le client est indiquée sous la forme :

« HTTP/x.y »

avec couramment x.y = 1.0 ou 1.1

Les autres lignes de l'en-tête correspondent à des champs optionnels permettant de spécifier des préférences (sur la langue souhaitée pour un document par exemple), etc.

2.2. Format des réponses HTTP fournies par le serveur.

Les réponses du serveur sont constituées de deux parties :

- Un en-tête contenant notamment une ligne de statut indiquant si la demande a pu être satisfaite. Divers champs d'en-tête fournissent des renseignements au client pour que celui puisse traiter le document qui suit.
- Un corps (quand la demande de service a pu être remplie) contenant le document demandé.

L'en-tête et le corps sont séparés par une ligne vide (i.e. les codes <CR><LF>). Chaque ligne de l'en-tête se termine par <CR><LF>.

Exemple de réponse à une requête :

On soumet au serveur Web `www.msi.unilim.fr` la demande de document suivante :

```
GET /~poulingeas/index.htm HTTP/1.0
```

Le serveur fournit alors la réponse suivante :

```
HTTP/1.1 200 OK
Date: Tue, 18 Jan 2005 15:25:00 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.1.2
Last-Modified: Thu, 04 Nov 2004 11:20:11 GMT
Content-Length: 361
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>
Site de Patrick Poulingeas
</title>
</head>
<frameset cols="18%,82%">
<frame src="choix.htm" name="index" frameborder="0" scrolling="yes"
noresize>
<frame src="present.htm" name="Debut" frameborder="0">
</frameset>
</html>
```

La ligne de statut de l'en-tête de la réponse se présente sous la forme suivante :

Version HTTP Code d'état Texte

On obtient ainsi des lignes de statut comme :

```
HTTP/1.1 200 OK
```

ou :

```
HTTP/1.1 404 Not Found
```

Le premier chiffre du code d'état retourné par le serveur correspond à une catégorie de réponse. On a les catégories suivantes :

Chiffre	Catégorie
1	Réservé.
2	Requête ou action exécutée avec succès.
3	Accès impossible, redirection nécessaire.
4	Requête invalide.
5	Erreur du serveur.

Examinons succinctement quelques codes de statut :

Code	Signification
200	Opération effectuée avec succès.
201	Envoyé à la suite d'un POST lorsque le document concerné a été établi.
202	Requête enregistrée ; elle sera exécutée plus tard.
204	Le serveur a traité la demande, mais il n'existe aucun document à renvoyer.
301	Le document réclamé a été déplacé et se trouve maintenant à une autre adresse mentionnée dans la réponse.
304	Réponse à une requête GET lorsque le serveur ne renvoie pas de document car il n'y a pas de nouvelle version de celui-ci.
305	Le document doit être accédé via un mandataire (<i>proxy</i>).
400	Erreur de syntaxe : le serveur n'a pas compris la requête.
401	Le client doit s'authentifier avant d'avoir accès au document demandé.
403	Accès au document interdit, même avec une authentification éventuelle.
404	Le serveur n'a pu trouver le document demandé.
405	Méthode non autorisée pour le document spécifié.
406	Le document n'existe pas dans le format exigé par le client.
500	Erreur interne du serveur.
501	Le serveur ne prend pas en charge l'action de la requête.
503	Le serveur est actuellement indisponible.

Le texte en fin de la ligne de statut est une information, liée au code d'état, destinée au client.

3. Les principales fonctionnalités fournies par HTTP.

Nous allons présenter les champs les plus importants qui peuvent apparaître dans l'en-tête d'une requête ou d'une réponse. Les noms de ces champs sont indifférents à la casse.

3.1. Codification du type de document.

Un client peut indiquer au serveur le type de document qu'il accepte en réponse. Inversement, le serveur indique au client le type du document qu'il lui envoie. Le type est défini avec la norme MIME (*Multipurpose Internet Mail Extension*).

- Du côté du client.

On dispose du champ `Accept` pour spécifier le type de document attendu. C'est un champ optionnel. S'il n'est pas présent dans la requête, le client accepte tout type de média comme réponse du serveur.

Habituellement, le navigateur déduit le type demandé à partir de l'extension du fichier apparaissant dans l'URL. Mais si le serveur a la possibilité de renvoyer un document dans différentes représentations (au sens d'un type MIME) – comme différents formats pour une même image –, le client peut exprimer sa préférence avec le champ `Accept`.

Plusieurs types peuvent être mentionnés dans une même ligne `Accept`, séparés par une virgule. C'est le type le plus précis qui est choisi de préférence par le serveur.

Exemple d'emploi de ce champ dans l'en-tête d'une requête :

`Accept: text/*, text/html, */*`

Le serveur essaie alors de renvoyer un document en suivant l'ordre de préférence suivant pour le type :

1. `text/html`
2. `text/*` → N'importe quel type de document texte
3. `*/*` → N'importe quel type de document

Si un champ `Accept` est mentionné dans une requête et que le serveur ne peut trouver aucun document correspondant au(x) type(s) demandé(s), il retourne le code d'erreur 406.

- Du côté du serveur.

Un champ `Content-Type` est **obligatoirement** renvoyé par le serveur dans sa réponse au client.

Le navigateur se base sur ce champ pour décider du traitement à accomplir sur le document renvoyé.

Le serveur détermine le type MIME d'un fichier en se basant sur son extension. Par exemple, Apache dispose à cette fin d'un fichier de correspondance entre extension et type MIME : `/etc/apache2/conf/mime.types`

Exemple d'emploi du champ `Content-Type` :

`Content-Type: text/html`

3.2. Sélection du jeu de caractères.

Le champ `Accept-Charset` dans une requête indique un jeu de caractères préféré pour la réponse. Pour pouvoir tenir compte de cette requête, le document doit exister sous différentes formes sur le serveur – à moins qu'il ne soit créé pour satisfaire la demande. Si le serveur ne peut pas trouver un document avec le jeu de caractères demandé, il renvoie l'erreur 406.

Remarque : selon les normes HTTP/1.0 et HTTP/1.1, le jeu de caractères associé par défaut aux sous-types du média « text » (i.e. `text/*`) est l'ISO-8859-1 (Latin1). En pratique, nombre d'anciens navigateurs de l'époque de HTTP/1.0 ne respectaient pas cette convention. La RFC 2616 détaillant la spécification HTTP/1.1 indique qu'un client HTTP/1.1 **doit** respecter une indication du jeu de caractères quand elle apparaît dans le champ `Content-Type` retourné par le serveur, comme par exemple :

`Content-Type: text/html; charset=iso-8859-1`

3.3. Sélection de la langue.

Le champ Accept-Language permet au client d'indiquer une préférence quant à la langue du document demandé.

Le champ Content-Language, retourné par le serveur, signale au client quelle est la langue du document qu'il lui fournit en réponse à sa requête.

Ces options n'ont d'intérêt que si le serveur dispose de plusieurs versions dans différentes langues du même document. Apache permet de distinguer ces versions en prenant en compte une extension du fichier correspondant à la langue (Utilisation de l'option Multiviews dans le fichier de configuration d'Apache). On peut donc rencontrer dans un même répertoire du serveur des fichiers du genre :

index.html.de	ou	index.de.html
index.html.fr	ou	index.fr.html
index.html.en	ou	index.en.html

Le premier fichier est la version allemande du document, le deuxième la version française, et le troisième la version anglaise.

3.4. Longueur du document envoyé.

Le champ Content-Length indique la taille en octets du document présent dans le corps du message. Ce champ peut apparaître dans une requête (envoi du contenu des champs d'un formulaire à un serveur par exemple) ou dans une réponse. Quand il n'apparaît pas dans une réponse, le corps du message est délimité par la clôture de la liaison. Dans une requête contenant un corps à l'attention du serveur, ce champ est **obligatoire**.

Exemple :

Content-Length: 361

3.5. Demande d'une nouvelle version d'un document.

Un client peut demander à recevoir un document en réponse à un GET uniquement si ce document a été modifié depuis une certaine date. Cette option a un intérêt quand le client gère un cache de pages Web (comme le fait, par exemple, un mandataire (*proxy*)).

Cette possibilité est obtenue en rajoutant le champ If-Modified-Since dans la requête.

Exemple :

If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT

Si le serveur ne dispose pas d'une version du document plus récente que la date et l'heure indiquées, il renvoie le code d'état 304.

3.6. Identification des correspondants.

Le client peut s'identifier auprès du serveur en employant le champ User-Agent dans sa requête. Ce champ permet au serveur d'établir des statistiques sur les navigateurs avec lesquels les internautes visitent ses sites. Il permet aussi au serveur de pouvoir adapter le contenu du document renvoyé aux capacités du navigateur utilisé par l'internaute.

Exemple d'utilisation du champ User-Agent :

User-Agent : links(0.83 ;)

Exemple de tableau généré par l'outil The Webalizer montrant des statistiques de fréquentation d'un serveur Web au cours du mois de mars 2002 :

Top 15 of 36 Total User Agents			
#	Hits		User Agent
1	174	19.48%	MSIE 5.0
2	172	19.26%	htdig/3.1.5 (root@localhost)
3	127	14.22%	MSIE 6.0
4	85	9.52%	Mozilla/4.7
5	68	7.61%	MSIE 5.5
6	57	6.38%	Mozilla/5.0
7	37	4.14%	Googlebot/2.1 (+http://www.googlebot.com/bot.html)
8	22	2.46%	Mozilla/4.5
9	22	2.46%	PabloElRobotEspa
10	16	1.79%	Konqueror/2.1
11	16	1.79%	Mozilla/3.0
12	12	1.34%	Indy Library)"
13	12	1.34%	htdig/3.1.5 (webmaster@bjsn.net)
14	11	1.23%	Wget/1.8.1
15	6	0.67%	MSIE 4.0

Le client peut indiquer au serveur l'origine de sa requête (typiquement à partir de quelle page Web il a été conduit par l'internaute à interroger le serveur). Ceci permet aux webmestres de vérifier quels sites référencent directement le leur.

Exemple d'utilisation du champ Referer :

Referer: <http://www.w3.org/hypertext/DataSources/Overview.html>

Le serveur peut aussi s'identifier auprès du client lors d'une réponse. Ceci s'effectue à l'aide du champ Server. Pour des raisons de sécurité ce champ n'est parfois pas retourné par le serveur Web.

Exemple d'utilisation du champ Server :

Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.1.2

3.7. Date et délai d'expiration.

Il est possible d'avoir une indication sur la date et l'heure où une opération s'est effectuée (pour une requête ou une réponse). Ceci est accompli à l'aide du champ Date. **Attention** : Dans une réponse d'un serveur, ce champ n'indique pas la date et l'heure de la dernière modification du document mais la date et l'heure de la réponse du serveur.

Exemple d'emploi du champ Date :

Date: Tue, 18 Jan 2005 15:25:00 GMT

C'est le champ Last-Modified qui, fourni par le serveur, renseigne le client sur la date et l'heure de la dernière modification du document renvoyé.

Exemple d'utilisation du champ Last-Modified dans une réponse :
Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT

4. Les cookies.

Un cookie est une simple association entre un nom et une valeur que le navigateur fournit automatiquement à un site après que celui-ci lui ait demandé d'enregistrer cette association. C'est donc le serveur Web qui décide de demander à un client de stocker des informations qui lui permettront de reconnaître l'internaute lors d'une prochaine visite.

La gestion d'un cookie se fait par ajout de lignes d'en-têtes au niveau HTTP.

Par exemple, après avoir rentré le nom « dupond » dans un formulaire, le serveur Web souhaite que cette information soit envoyée systématiquement quand la personne naviguera sur le site (afin qu'il n'ait pas à remplir de nouveau le formulaire pour s'identifier). Quand l'internaute soumet le formulaire au serveur, celui-ci envoie dans la réponse HTTP au client le champ d'en-tête suivant :

```
Set-Cookie2: client="dupond"; Version="1"; Path="/";  
Expires Fri, 11-Feb-2005 14:00:00 GMT
```

Cette ligne demande au navigateur le stockage d'un cookie dont :

- le nom est « client » et la valeur est « dupond ».
- le chemin d'accès est « / », soit le site dans sa globalité.
- la date d'expiration est le 11 février 2005 à 14h, heure de Greenwich.

Lors d'une nouvelle requête au serveur Web, le navigateur Web ajoute alors automatiquement la ligne suivante dans son en-tête HTTP :

```
Cookie: $Version="1"; client="dupond"; $Path="/"
```

Un cookie est renvoyé par le navigateur quand on demande une ressource placée dans l'arborescence du chemin d'accès spécifié plus tôt par le serveur (attribut « Path » de Set-Cookie2).

Exemple :

Le serveur www.foo.org a renvoyé les champs d'en-tête suivants au navigateur:

```
Set-Cookie2: Nom="Toto"; Version="1"; Path="/doc"
```

```
Set-Cookie2: NomPrive="TotoPrive"; Version="1"; Path="/doc/private"
```

Par la suite, si le navigateur demande la ressource www.foo.org/doc/private/texte.php, il fournit dans sa requête au serveur le champ d'en-tête suivant :

```
Cookie: $Version="1"; NomPrive="TotoPrive"; $Path="/doc/private";  
Nom="Toto"; $Path="/doc"
```

On voit clairement sur cet exemple qu'un cookie se définit donc plus précisément comme une association entre un nom et une valeur + un chemin (Nom="Toto" apparaît dans la requête construite par le navigateur parce que l'URL contient la chaîne « www.foo.org/doc »).

Si le navigateur demande le document `www.foo.org/doc/manuel.php`, il ajoute alors uniquement dans sa requête la ligne suivante :

`Cookie: $Version="1"; Nom="Toto"; $Path="/doc"`

Dans les exemples ci-dessus, on remarque qu'il n'y a aucun cryptage par défaut de la valeur du cookie (que ce soit quand le serveur le crée ou quand le navigateur le renvoie). On a donc besoin de disposer d'une connexion sécurisée pour utiliser un cookie contenant des informations « sensibles ». L'attribut « Secure » qui peut apparaître dans une ligne d'en-tête `Set-Cookie2` indique au navigateur qu'il doit utiliser une liaison sécurisée quand il renvoie le cookie au serveur.

Quelques remarques :

- Plusieurs directives `Set-Cookie2` peuvent être retournées par un même serveur.
- Un navigateur devrait satisfaire les exigences suffisantes :
 - pouvoir mémoriser au minimum 300 cookies,
 - avoir une taille minimale de 4000 octets pour un cookie,
 - conserver au minimum 20 cookies par serveur.
- Un cookie peut être effacé de la machine du client en indiquant pour l'attribut « Expires » une date expirée.
- Un cookie envoyé sans date par le serveur expire quand l'internaute ferme son navigateur.

Les cookies sont stockés par le navigateur dans un fichier ou un répertoire sur la machine du client. Par exemple, sous Windows XP, Internet Explorer 6.0 conserve ceux-ci à l'aide de fichiers textes placés dans les répertoires :

`C:\Documents and Settings\nom_utilisateur\Cookies`

`C:\Documents and Settings\nom_utilisateur\Local Settings\Temporary Internet Files`

Le dernier standard concernant la gestion des cookies a été publié par l'IETF dans la RFC 2965.