Équipe d'ingénierie de l'Internet (IETF)
**Request for Comments : 6265**
RFC rendue obsolète : 2965
Catégorie : Sur la voie de la normalisation
ISSN: 2070-1721

A. Barth, U.C. Berkeley
avril 2011

Traduction Claude Brière de L'Isle

# Mécanisme de gestion d'état HTTP

**Résumé**
Le présent document définit les champs d'en-tête HTTP Cookie et Set-Cookie. Ces champs d'en-tête peuvent être utilisés par les serveurs HTTP pour mémoriser l'état (appelé cookie) chez les agents d'utilisateur HTTP, laissant les serveurs conserver une session à états pleins sur le protocole HTTP presque sans état. Bien que les cookies aient eu historiquement de nombreux malheurs qui dégradent leur sécurité et leur confidentialité, les champs d'en-tête Cookie et Set-Cookie sont largement utilisés sur l'Internet. Le présent document rend obsolète la RFC 2965.



## Table des matières

## 1. Introduction

Le présent document définit les champs d'en-tête HTTP Cookie et Set-Cookie. En utilisant le champ d'en-tête Set-Cookie, un serveur HTTP peut passer des paires de nom/valeur et leurs métadonnées associées (appelées des cookies) à un agent d'utilisateur. Quand l''agent d'utilisateur fait des demandes ultérieures au serveur, il utilise les métadonnées et d'autres informations pour déterminer si il faut retourner les paires de nom/valeur dans l'en-tête Cookie.

Bien que simple en apparence, les cookies ont un certain nombre de complexités. Par exemple, le serveur indique une portée pour chaque cookie quand il l'envoie à l'agent d'utilisateur. La portée indique la durée maximum pendant laquelle l'agent d'utilisateur devrait retourner le cookie, les serveurs auxquels l'agent d'utilisateur devrait retourner le cookie, et les schémas d'URI pour lesquels le cookie est applicable.

Pour des raisons historiques, les cookies contiennent un certain nombre de défauts de sécurité et de confidentialité. Par exemple, un serveur peut indiquer qu'un certain cookie est destiné à des connexions "sûres", mais l'attribut Secure n'assure pas l'intégrité en présence d'un attaquant actif du réseau. De même, les cookies pour un certain hôte sont partagés sur tous les accès de cet hôte, même si la "politique de même origine" usuelle utilisée par les navigateurs de la Toile isole les contenus restitués via les différents accès.

La présente spécification s'adresse à deux publics : les développeurs de serveurs qui génèrent des cookies, et les développeurs d'agents d'utilisateur consommateurs de cookies.

Pour maximiser l'interopérabilité avec les agents d'utilisateur, les serveurs DEVRAIENT se limiter au profil de bon comportement défini à la Section 4 lorsque ils génèrent des cookies.

Les agents d'utilisateur DOIVENT mettre en œuvre les règles de traitement les plus libérales définies à la Section 5, afin de maximiser l'interopérabilité avec les serveurs existants qui ne se conforment pas au profil de bon comportement défini à la Section 4.

Le présent document spécifie la syntaxe et la sémantique de ces en-têtes comme ils sont actuellement utilisés sur l'Internet.

En particulier, le présent document ne crée pas une nouvelle syntaxe ou sémantique différentes de celles en usage aujourd'hui. Les recommandations pour la génération de cookies données à la Section 4 représentent un sous ensemble préféré du comportement actuel de serveur, et même le plus libéral des algorithmes de traitement de cookie fourni à la Section 5 ne recommande pas toutes les variantes syntaxiques et sémantiques en usage aujourd'hui. Lorsque un logiciel existant diffère du protocole recommandé de façon significative, le document contient une note expliquant la différence.

Avant le présent document, il y avait au moins trois descriptions de cookies : la "spécification des cookies Netscape" [Netscape], la [RFC2109], et la [RFC2965]. Cependant, aucun de ces documents ne décrit comment les en-têtes Cookie et Set-Cookie sont en fait utilisés sur l'Internet (voir [Kri2001] pour le contexte historique). En relation avec les spécifications précédentes de l'IETF des mécanismes de gestion d'état HTTP, le présent document demande les actions suivantes :
1. Changer le statut de la [RFC2109] en Historique (elle a déjà été rendue obsolète par la [RFC2965]).
2. Changer le statut de la [RFC2965] en Historique.
3. Indiquer que la [RFC2965] a été rendue obsolète par le présent document.

En particulier, en passant la RFC 2965 au statut de Historique et en la rendant obsolète, le présent document déconseille l'utilisation des champs d'en-tête Cookie2 et Set-Cookie2.

## 2. Conventions

### 2.1 Critères de conformité

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Les exigences formulées à l'impératif au titre des algorithmes (comme "supprimer tout caractère espace en tête" ou "retourner faux et interrompre ces étapes") sont à interpréter avec la signification des mots clés "DOIT", "DEVRAIT", "PEUT", etc. utilisés en introduisant l'algorithme.

Les exigences de conformité formulées comme des algorithmes ou des étapes spécifiques peuvent être mises en œuvre de n'importe quelle manière, pour autant que le résultat final soit équivalent. En particulier, les algorithmes définis dans cette spécification sont destinés à être faciles à comprendre et non à être performants.

### 2.2 Notation de la syntaxe

Le cœur des règles suivant est inclus par référence, comme défini dans la [RFC5234], Appendice B.1 : ALPHA (lettres), CR (retour chariot), CRLF (retour chariot saut à la ligne), CTL (caractères de contrôle), DIGIT (chiffres de 0 à 9), DQUOTE (guillemets), HEXDIG (chiffres hexadécimaux 0-9/A-F/a-f), LF (saut à la ligne), OCTET (toute séquence de données de huit bits), SP (espace), HTAB (tabulation horizontale), CHAR (tout caractère US-ASCII), VCHAR (tout caractère US-ASCII visible), et WSP (espace blanche).

La règle OWS est utilisée lorsque zéro, un ou plusieurs octets d'espace blanche linéaire PEUVENT apparaître :

```
OWS         = *( [ obs-fold ] WSP )                ; espace blanche "facultative"
obs-fold    = CRLF
```

Les OWS DEVRAIENT soit n'être pas produits, soit être produits comme un seul SP.

### 2.3 Terminologie

Les termes d'agent d'utilisateur, client, serveur, mandataire, et serveur d'origine ont la même signification que dans la spécification HTTP/1.1 ([RFC2616], paragraphe 1.3).

L'hôte de demande est le nom de l'hôte, tel que connu par l'agent d'utilisateur, auquel l'agent d'utilisateur envoie une demande HTTP ou duquel il reçoit une réponse HTTP (c'est-à-dire, le nom le l'hôte auquel il envoie la demande HTTP correspondante).

Le terme URI de demande est défini au paragraphe 5.1.2 de la [RFC2616].

Deux séquences d'octets sont dites correspondre indépendamment de la casse l'une à l'autre si et seulement si elles sont équivalentes selon le collationnement i;ascii-casemap défini dans la [RFC4790].

Le terme chaîne signifie une séquence d'octets non NULS.

## 3. Généralités

Cette section présente un moyen pour qu'un serveur d'origine envoie des informations d'état à un agent d'utilisateur et pour que l'agent d'utilisateur retourne les informations d'état au serveur d'origine.

Pour mémoriser l'état, le serveur d'origine inclut un en-tête Set-Cookie dans une réponse HTTP. Dans les demandes suivantes, l'agent d'utilisateur retourne un en-tête Demande de Cookie au serveur d'origine. L'en-tête Cookie contient les cookies que l'agent d'utilisateur a reçu dans les en-têtes Set-Cookie précédents. Le serveur d'origine est libre d'ignorer l'en-tête Cookie ou d'utiliser son contenu pour des besoins définis par l'application.

Les serveurs d'origine PEUVENT envoyer un en-tête de réponse Set-Cookie avec toute réponse. Les agents d'utilisateur PEUVENT ignorer les en-têtes Set-Cookie contenus dans des réponses avec un code d'état de niveau 100, mais DOIVENT traiter les en-têtes Set-Cookie contenus dans les autres réponses (incluant les réponses avec des codes d'état de niveau 400 et 500). Un serveur d'origine peut inclure plusieurs champs d'en-tête Set-Cookie dans une seule réponse. La présence d'un Cookie ou d'un champ d'en-tête Set-Cookie n'empêche pas les antémémoires HTTP de mémoriser et réutiliser une réponse.

Les serveurs d'origine NE DEVRAIENT PAS placer plusieurs champs d'en-tête Set-Cookie dans un seul champ d'en-tête. Le mécanisme usuel pour placer les champs d'en-têtes HTTP (c'est-à-dire, comme défini dans la [RFC2616]) peut changer la sémantique du champ d'en-tête Set-Cookie parce que le caractère %x2C (",") est utilisé par Set-Cookie d'une façon qui entre en conflit avec ce placement.

### 3.1 Exemples

En utilisant l'en-tête Set-Cookie, un serveur peut envoyer à l'agent d'utilisateur une courte chaîne dans une réponse HTTP que l'agent d'utilisateur retournera dans les futures demandes HTTP qui sont dans la portée du cookie. Par exemple, le serveur peut envoyer à l'agent d'utilisateur un "identifiant de session" nommé SID avec la valeur 31d4d96e407aad42. L'agent d'utilisateur retourne alors l'identifiant de session dans les demandes suivantes.

== Serveur -> Agent d'utilisateur ==

Set-Cookie: SID=31d4d96e407aad42

== Agent d'utilisateur -> Serveur ==

Cookie: SID=31d4d96e407aad42

Le serveur peut altérer la portée par défaut du cookie en utilisant les attributs Path et Domain. Par exemple, le serveur peut donner pour instruction à l'agent d'utilisateur de retourner le cookie à chaque chemin et pour tous les sous domaines de exemple.com.

== Serveur -> Agent d'utilisateur ==

Set-Cookie: SID=31d4d96e407aad42; Path=/; Domain=exemple.com

== Agent d'utilisateur -> Serveur ==

Cookie: SID=31d4d96e407aad42

Comme le montre l'exemple suivant, le serveur peut mémoriser plusieurs cookies chez l'agent d'utilisateur. Par exemple, le serveur peut mémoriser un identifiant de session aussi bien que le langage préféré de l'utilisateur en retournant deux champs d'en-tête Set-Cookie. On note que le serveur utilise les attributs Secure et HttpOnly pour fournir des protections de sécurité supplémentaires pour un identifiant de session plus sensible (voir au paragraphe 4.1.2.)

== Serveur -> Agent d'utilisateur ==

Set-Cookie: SID=31d4d96e407aad42; Path=/; Secure; HttpOnly
Set-Cookie: lang=en-US; Path=/; Domain=example.com

== Agent d'utilisateur -> Serveur ==

Cookie: SID=31d4d96e407aad42; lang=en-US

On note que l'en-tête Cookie ci-dessus contient deux cookies, l'un appelé SID et l'autre nommé lang. Si le serveur souhaite que l'agent d'utilisateur conserve le cookie sur plusieurs "sessions" (par exemple, quand agent d'utilisateur redémarre) le serveur peut spécifier une date d'expiration dans l'attribut Expires. Noter que l'agent d'utilisateur pourrait supprimer le cookie avant la date d'expiration si la mémorisation de cookies de l'agent d'utilisateur excède son quota ou si l'utilisateur supprime manuellement le cookie du serveur.

== Serveur -> Agent d'utilisateur ==

Set-Cookie: lang=en-US; Expires=Wed, 09 Jun 2021 10:18:14 GMT

== Agent d'utilisateur -> Serveur ==

Cookie: SID=31d4d96e407aad42; lang=en-US

Finalement, pour supprimer un cookie, le serveur retourne un en-tête Set-Cookie avec une date d'expiration dans le passé. Le serveur n'aura réussi à supprimer le cookie que si le chemin et l'attribut Domaine dans l'en-tête Set-Cookie correspondent aux valeurs utilisées à la création du cookie.

== Serveur -> Agent d'utilisateur ==

Set-Cookie: lang=; Expires=Sun, 06 Nov 1994 08:49:37 GMT

== Agent d'utilisateur -> Serveur ==

Cookie: SID=31d4d96e407aad42

## 4. Exigences pour le serveur

Cette section décrit la syntaxe et la sémantique d'un profil de bon comportement des en-têtes Cookie et Set-cookie.

### 4.1 Set-Cookie

L'en-tête HTTP de réponse Set-Cookie est utilisé pour envoyer des cookies du serveur à l'agent d'utilisateur.

#### 4.1.1 Syntaxe

Informellement, l'en-tête de réponse Set-Cookie contient le nom d'en-tête "Set-Cookie" suivi par un ":" et un cookie. Chaque cookie commence par une paire nom-valeur, suivie par zéro, une ou plusieurs paires attribut-valeur. Les serveurs NE DEVRAIENT PAS envoyer des en-têtes Set-Cookie qui ne se conforment pas à la grammaire suivante :

```
en-tête-set-cookie  = "Set-Cookie:" SP chaîne-set-cookie
chaîne-set-cookie  = paire-cookie *( ";" SP cookie-av )
paire-cookie  = nom-de-cookie "=" valeur-de-cookie
nom-de-cookie   = jeton
valeur-de-cookie  = *cookie-octet / ( DQUOTE *cookie-octet DQUOTE )
cookie-octet  = %x21 / %x23-2B / %x2D-3A / %x3C-5B / %x5D-7E
          ; caractères US-ASCII à l'exclusion des CTL, espaces, guillemets, virgule, point-virgule, et barre oblique inverse
jeton  = <token, défini dans la [RFC2616], paragraphe 2.2>

cookie-av  = expires-av / max-age-av / domain-av / path-av / secure-av / httponly-av / extension-av
expires-av  = "Expires=" sane-cookie-date
sane-cookie-date  = <date-rfc1123, définie dans la [RFC2616], paragraphe 3.3.1>
max-age-av  = "Max-Age=" chiffre-non-zéro *CHIFFRE
          ; En pratique expires-av et max-age-av sont tous deux limités aux dates représentables par l'agent d'utilisateur.
chiffre-non-zéro  = %x31-39                    ; chiffres de 1 à 9
```

domain-av = "Domain=" valeur-de-domaine
valeur-de-domaine = <sous(domaine>
; défini dans la [RFC1034], paragraphe 3.5, amendé par la [RFC1123], paragraphe 2.1
path-av = "Path=" valeur-de-chemin
valeur-de-chemin = <tout caractère sauf les CTL ou ";">
secure-av = "Secure"
httponly-av = "HttpOnly"
extension-av = <tout caractère sauf les CTL ou ";">

Noter que certains des termes de la grammaire ci-dessus se réfèrent à des documents qui utilisent une notation grammaticale différente de celle du présent document (qui utilise l'ABNF de la [RFC5234]).

La sémantique de valeur-de-cookie n'est pas définie dans le présent document.

Pour maximiser la compatibilité avec les agents d'utilisateur, les serveurs qui souhaitent mémoriser des données arbitraires dans une valeur-de-cookie DEVRAIENT coder ces données, par exemple, en utilisant le Base64 [RFC4648].

Les portions de la chaîne-set-cookie produites par le terme cookie-av sont appelées des attributs. Pour maximiser la compatibilité avec les agents d'utilisateur, les serveurs NE DEVRAIENT PAS produire deux attributs avec le même nom dans la même chaîne-set-cookie. (Voir au paragraphe 5.3 comment les agents d'utilisateur traitent ce cas.)

Les serveurs NE DEVRAIENT PAS inclure plus d'un champ d'en-tête Set-Cookie dans la même réponse avec le même nom-de-cookie. (Voir au paragraphe 5.2 comment les agents d'utilisateur traitent ce cas.)

Si un serveur envoie concurremment plusieurs réponses contenant des en-têtes Set-Cookie à l'agent d'utilisateur (par exemple, quand il communique avec l'agent d'utilisateur sur plusieurs prises) ces réponses créent une "condition de compétition" qui peut conduire à un comportement imprévisible.

Note : Certains agents d'utilisateur existants diffèrent dans leur interprétation d'une année à deux chiffres. Pour éviter les problèmes de compatibilité, les serveurs DEVRAIENT utiliser le format date de la rfc1123, qui exige une année de quatre chiffres.

Note : Certains agents d'utilisateur mémorisent et traitent les dates dans les cookies comme des valeurs UNIX time_t de 32 bits. Des erreurs de mise en œuvre dans les bibliothèques qui prennent en charge le traitement de time_t sur certains systèmes pourraient être cause que de tels agents d'utilisateur traitent de façon incorrecte les dates après 2038.

### 4.1.2 Sémantique (non normative)

Ce paragraphe décrit de façon simplifiée la sémantique de l'en-tête Set-Cookie. Cette sémantique est assez détaillée pour la compréhension de la plupart des cookies les plus courants utilisés par les serveurs. Le sémantique complète est décrite à la Section 5.

Quand l'agent d'utilisateur reçoit un en-tête Set-Cookie, il mémorise le cookie avec ses attributs. Ensuite, quand l'agent d'utilisateur fait une demande HTTP, il inclut les cookies applicables non expirés dans l'en-tête Cookie.

Si l'agent d'utilisateur reçoit un nouveau cookie avec le même nom-de-cookie, valeur-de-domaine, et valeur-de-chemin qu'un cookie qu'il déjà mémorisé, le cookie existant est évincé et remplacé par le nouveau. Remarquer que les serveurs peuvent supprimer les cookies en envoyant à l'agent d'utilisateur un nouveau cookie avec un attribut Expires qui a une valeur dans le passé.

Sauf si les attributs du cookie indiquent le contraire, le cookie n'est retourné que au serveur d'origine (et non, par exemple, à un sous domaine) et il expire à la fin de la session en cours (comme défini par l'agent d'utilisateur). Les agents d'utilisateurs ignorent les attributs de cookie non reconnus (mais pas tout le cookie).

#### 4.1.2.1 Attribut Expires

L'attribut Expires indique la durée de vie maximum du cookie, représentée par la date et l'heure d'expiration du cookie. L'agent d'utilisateur n'est pas obligé de conserver le cookie jusqu'à l'expiration de la date spécifiée. En fait, les agents d'utilisateur évincent souvent les cookies à cause de la pression sur la mémoire ou des soucis de confidentialité.

#### 4.1.2.2 Attribut Max-Age

L'attribut Max-Age indique la durée de vie maximum du cookie, représenté comme le nombre de secondes jusqu'à l'expiration du cookie. L'agent d'utilisateur n'est pas obligé de conserver le cookie pour la durée spécifiée. En fait, les agents d'utilisateur évincent souvent les cookies à cause de la pression sur la mémoire ou des soucis de confidentialité.

Note : Certains agents d'utilisateur existants ne prennent pas en charge l'attribut Max-Age. Les agents d'utilisateur qui ne prennent pas en charge l'attribut Max-Age ignorent l'attribut.

Si un cookie a les deux attributs Max-Age et Expires, l'attribut Max-Age a la préséance et contrôle la date d'expiration du cookie. Si un cookie n'a ni l'attribut Max-Age ni l'attribut Expires, l'agent d'utilisateur va conserver le cookie jusqu'à la fin de la session en cours (comme défini par l'agent d'utilisateur).

#### 4.1.2.3 Attribut Domain

L'attribut Domain spécifie les hôtes auxquels le cookie sera envoyé. Par exemple, si la valeur de l'attribut Domain est "exemple.com", l'agent d'utilisateur va inclure le cookie dans l'en-tête Cookie quand il fait des demandes HTTP à exemple.com, www.exemple.com, et www.corp.exemple.com. (Noter qu'un caractère %x2E (".") en tête, si il est présent, est ignoré même si ce caractère n'est pas permis, mais le même, s'il est présent en queue va faire que l'agent d'utilisateur ignorera l'attribut.) Si le serveur omet l'attribut Domain, l'agent d'utilisateur ne retournera le cookie qu'au serveur d'origine.

Avertissement : Certains agents d'utilisateur traitent un attribut Domain absent comme si l'attribut Domain était présent et contenait le nom de l'hôte actuel. Par exemple, si exemple.com retourne un en-tête Set-Cookie sans attribut Domain, ces agents d'utilisateur vont à tort envoyer aussi le cookie à www.exemple.com.

L'agent d'utilisateur va rejeter les cookies sauf si l'attribut Domain spécifie une portée pour le cookie qui va inclure le serveur d'origine. Par exemple, l'agent d'utilisateur va accepter un cookie avec un attribut Domain de "exemple.com" ou de "foo.exemple.com" provenant de foo.exemple.com, mais l'agent d'utilisateur ne va pas accepter un cookie avec un attribut Domain de "bar.exemple.com" ou de "baz.foo.exemple.com".

Note : pour des raisons de sécurité, de nombreux agents d'utilisateur sont configurés à rejeter les attributs Domain qui correspondent à des "suffixes publics". Par exemple, certains agents d'utilisateur vont rejeter les attributs Domain de "com" ou "co.uk". (Voir plus d'informations au paragraphe 5.3.)

#### 4.1.2.4 Attribut Path

La portée de chaque cookie est limitée à un ensemble de chemins, contrôlés par l'attribut Path. Si le serveur omet l'attribut Path, l'agent d'utilisateur va utiliser le "répertoire" du composant de chemin de l'uri-de-demande comme valeur par défaut. Voir plus de détails au paragraphe 5.1.4.)

L'agent d'utilisateur ne va inclure le cookie dans une demande HTTP que si la portion de chemin de l'uri-de-demande correspond (ou est un sous répertoire) à l'attribut Path du cookie, où le caractère %x2F ("/") est interprété comme séparateur de répertoire.

Bien qu'apparemment utile pour isoler les cookies entre des chemins différents au sein d'un certain hôte, on ne peut pas s'appuyer sur l'attribut Path pour la sécurité (voir à la Section 8).

#### 4.1.2.5 Attribut Secure

L'attribut Secure limite la portée du cookie aux canaux "sûrs" (où "sûr" est défini par l'agent d'utilisateur). Quand un cookie a l'attribut Secure, l'agent d'utilisateur ne va inclure le cookie dans une demande HTTP que si la demande est transmise sur un canal sûr (normalement HTTP sur la sécurité de la couche transport (TLS, *Transport Layer Security*) [RFC2818]).

Bien qu'apparemment utile pour protéger les cookies contre les attaquants actifs du réseau, l'attribut Secure ne protège que la confidentialité du cookie. Un attaquant actif du réseau peut écraser les cookies Secure d'un canal non sûr, perturbant leur intégrité (voir plus de détails au paragraphe 8.6).

#### 4.1.2.6 Attribut HttpOnly

L'attribut HttpOnly *(seulement HTTP)* limite la portée du cookie aux demandes HTTP. En particulier, l'attribut donne pour instruction à l'agent d'utilisateur d'omettre le cookie quand il fournit l'accès aux cookies via des API "non HTTP" (comme une API de navigateur qui expose les cookies à des scripts).

Noter que l'attribut HttpOnly est indépendant de l'attribut Secure : un cookie peut avoir les deux attributs HttpOnly et Secure.

### 4.2 Cookie

#### 4.2.1 Syntaxe

L'agent d'utilisateur envoie les cookies mémorisés au serveur d'origine dans l'en-tête Cookie. Si le serveur se conforme aux exigences du paragraphe 4.1 (et si l'agent d'utilisateur se conforme aux exigences de la Section 5) l'agent d'utilisateur va envoyer un en-tête Cookie qui se conforme à la grammaire suivante :

en-tête-cookie = "Cookie:" OWS chaîne-cookie OWS
chaîne-cookie = paire-cookie *( ";" SP paire-cookie )

#### 4.2.2 Sémantique

Chaque paire-cookie représente un cookie mémorisé par l'agent d'utilisateur. La paire-cookie contient le nom-de-cookie et la valeur-de-cookie que l'agent d'utilisateur a reçus dans l'en-tête Set-Cookie.

On note que les attributs de cookie ne sont pas retournés. En particulier, le serveur ne peut pas déterminer d'après l'en-tête Cookie seul quand un cookie va arriver à expiration, pour quels hôtes le cookie est valide, pour quels chemins le cookie est valide, ou si le cookie a été établi avec les attributs Secure ou HttpOnly.

La sémantique des cookies individuels dans l'en-tête Cookie n'est pas définie par le présent document. Les serveurs sont supposés attribuer à ces cookies une sémantique spécifique de l'application.

Bien que les cookies soient mis en série linéaire dans l'en-tête Cookie, les serveurs NE DEVRAIENT PAS se fier à l'ordre de présentation. En particulier, si l'en-tête Cookie contient deux cookies du même nom (par exemple, ils ont été établis avec des attributs Path ou Domain différents) les serveurs NE DEVRAIENT PAS se fier à l'ordre dans lequel ces cookies apparaissent dans l'en-tête.

## 5. Exigences pour l'agent d'utilisateur

Cette section spécifie les en-têtes Cookie et Set-Cookie en détail suffisant pour qu'un agent d'utilisateur qui met en œuvre les exigences avec précision puisse interopérer avec les serveurs existants (même ceux qui ne se conforment pas au profil de bon comportement décrit à la Section 4).

Un agent d'utilisateur pourrait appliquer plus de restrictions que celles spécifiées ici (par exemple, pour une sécurité améliorée) ; cependant, l'expérience a montré que trop de rigueur réduit la probabilité qu'un agent d'utilisateur soit capable d'interopérer avec les serveurs existants.

### 5.1 Algorithmes de sous composant

Ce paragraphe définit des algorithmes utilisés par les agents d'utilisateur pour traiter les sous composants spécifiques des en-têtes Cookie et Set-Cookie.

### 5.1.1 Dates

L'agent d'utilisateur DOIT utiliser un algorithme équivalent à l'algorithme suivant pour analyser une cookie-date. Noter que les divers fanions booléens définis au titre de l'algorithme (c'est-à-dire, trouve-l'heure, trouve-le-jour-du-mois, trouve-le-mois, trouve-l'an) ne sont initialement "pas établis".

1. En utilisant la grammaire ci-dessous, diviser la cookie-date en jetons-de-date

```
cookie-date  = *délimiteur liste-de-jetons-de-date *délimiteur
liste-de-jetons-de-date = jeton-de-date *( 1*délimiteur jeton-de-date )
jeton-de-date  = 1*non-délimiteur
délimiteur = %x09 / %x20-2F / %x3B-40 / %x5B-60 / %x7B-7E
non-délimiteur = %x00-08 / %x0A-1F / CHIFFRE / ":" / ALPHA / %x7F-FF
non-chiffre  = %x00-2F / %x3A-FF
jour-du-mois   = 1*2 CHIFFRE ( non-chiffre *OCTET )
mois   = ( "jan" / "fev" / "mar" / "avr" / "mai" / "jun" / "jul" / "aut" / "sep" / "oct" / "nov" / "dec" ) *OCTET
an  = 2*4 CHIFFRE ( non-chiffre *OCTET )
heure  = heure-hms ( non-chiffre *OCTET )
heure-hms  = champ-heure ":" champ-heure ":"  champ-heure
champ-heure   = 1*2 CHIFFRE
```

2. Traiter chaque jeton-de-date à la suite dans l'ordre où jeton-de-date apparaît dans la cookie-date :
   1. Si le fanion trouve-l'heure n'est pas établi et si le jeton correspond à l'heure de production, établir le fanion trouve-l'heure et régler valeur-de-l'heure, valeur-de-minute, et valeur-de-seconde aux valeurs notées par les chiffres dans le jeton-de-date, respectivement. Sauter les sous étapes suivantes et continuer avec le prochain jeton-de-date.
   2. Si le fanion trouve-le-jour-du-mois n'est pas établi et si le jeton-de-date correspond au jour du mois de production, établir le fanion trouve-le-jour-du-mois et régler la valeur-du-jour-du-mois au nombre noté par le jeton-de-date. sauter les sous étapes restantes et continuer avec le prochain jeton-de-date.
   3. Si le fanion trouve-le-mois n'est pas établi et si le jeton-de-date correspond au mois de production, établir le fanion trouve-le-mois et régler la valeur-du-mois au mois noté par le jeton-de-date. Sauter les sous étapes restantes et continuer avec le prochain jeton-de-date.
   4. Si le fanion trouve-l'an n'est pas établi et si le jeton-de-date correspond à l'année de production, établir le fanion trouve-l'an et régler la valeur-de-l'an au nombre noté par le jeton-de-date. Sauter les sous étapes restantes et continuer au prochain jeton-de-date.
3. Si la valeur-de-l'an est supérieure ou égale à 70 et inférieure ou égale à 99, incrémenter valeur-de-l'an de 1900.
4. Si valeur-de-l'an est supérieure ou égale à 0 et inférieure ou égale à 69, incrémenter valeur-de-l'an de 2000.
   Note : certains agents d'utilisateur existants interprètent les années à deux chiffres différemment.
5. Interrompre les étapes et échouer l'analyse de cookie-date si :
   * au moins un des fanions trouve-l'heure, trouve-le-jour-du-mois, trouve-le-mois, ou trouve-l'an n'est pas établi,
   * la valeur-du-jour-du mois est inférieure à 1 ou supérieure à 31,
   * la valeur-de-l'an est inférieure à 1601,
   * la valeur-de-l'heure est supérieure à 23,
   * la valeur-de-minute est supérieure à 59, ou
   * la valeur-de-seconde est supérieure à 59.
   (Noter que les secondes sautées e peuvent pas être représentées dans cette syntaxe.)
6. Soit date-de-cookie-analysée la date dont le jour du mois, le mois, l'année, l'heure, la minute, et la seconde (en UTC) sont respectivement la valeur-du-jour-du-mois, la valeur-du-mois, la valeur-de-l'an, la valeur-de-l'heure, la valeur-de-minute, et la valeur-de seconde. Si une telle date n'existe pas, interrompre ces étapes et échouer l'analyse de la cookie-date.
7. Retourner la date-de-cookie-analysée comme résultat de l'algorithme.

### 5.1.2 Noms d'hôtes canonisés

Un nom d'hôte canonisé est la chaîne générée par l'algorithme suivant :

1. Convertir le nom de l'hôte en une séquence d'étiquettes de noms de domaine individuels.

2. Convertir chaque étiquette qui n'est pas une étiquette LDH non réservée (NR-LDH) en une étiquette A (voir le paragraphe 2.3.2.1 de la [RFC5890] sur les deux définitions) ou en une "étiquette punycode" (étiquette résultant de la conversion "ToASCII" de la Section 4 de la [RFC3490]) comme approprié (voir au paragraphe 6.3 de la présente spécification).

3. Enchaîner les étiquettes résultantes, séparées par un caractère %x2E (".").

### 5.1.3 Confrontation de domaines

Une chaîne de domaine correspond à une certaine chaîne de domaine si au moins une des conditions suivantes est satisfaite :

- La chaîne de domaine et la chaîne sont identiques. (Noter que les deux chaînes de domaines auront été canonisées en minuscules à ce moment.)
- Toutes les conditions suivantes sont satisfaites :
  - La chaîne de domaine est un suffixe de la chaîne.
  - Le dernier caractère de la chaîne qui n'est pas inclus dans la chaîne de domaine est un caractère %x2E (".").
  - La chaîne est un nom d'hôte (c'est-à-dire, pas une adresse IP).

### 5.1.4 Chemins et correspondance de chemin

L'agent d'utilisateur DOIT utiliser un algorithme équivalent à l'algorithme suivant pour calculer le chemin par défaut d'un cookie :

1. Soit chemin-d'uri la portion de chemin de l'uri-de-demande si une telle portion existe (et vide autrement). Par exemple, si l'uri-de-demande contient juste un chemin (et une chaîne d'interrogation facultative) alors le chemin-d'uri est ce chemin (sans le caractère %x3F ("?") ou la chaîne d'interrogation) et si l'uri-de-demande contient un URI absolu complet, le chemin-d'uri est le composant path de cet URI.

2. Si le chemin-d'uri est vide ou si le premier caractère de chemin-d'uri n'est par un caractère %x2F ("/") produire un %x2F ("/") et sauter les étapes restantes.

3. Si le chemin-d'uri ne contient rien d'autre qu'un caractère %x2F ("/") produire un %x2F ("/") et sauter le reste de l'étape.

4. Sortir les caractères du chemin-d'uri depuis le premier caractère jusqu'au, non inclus, %x2F ("/") le plus à droite.

Un chemin de demande correspond au chemin d'un certain cookie si au moins une des conditions suivantes est satisfaite :

- Le chemin-de-cookie et le chemin-de-demande sont identiques.
- Le chemin-de-cookie est un préfixe du chemin-de-demande, et le dernier caractère du chemin-de-cookie est %x2F ("/").
- Le chemin-de-cookie est un préfixe du chemin-de-demande, et le premier caractère du chemin-de-demande qui n'est pas inclus dans le chemin-de-cookie est un caractère %x2F ("/").

## 5.2 En-tête Set-Cookie

Quand un agent d'utilisateur reçoit un champ d'en-tête Set-Cookie dans une réponse HTTP, l'agent d'utilisateur PEUT ignorer le champ d'en-tête Set-Cookie en totalité. Par exemple, l'agent d'utilisateur peut souhaiter bloquer les réponses aux demandes du "tiers" d'établir des cookies (voir le paragraphe 7.1).

Si l'agent d'utilisateur n'ignore pas le champ d'en-tête Set-Cookie dans sa totalité, il DOIT analyser la valeur-de-champ du champ d'en-tête Set-Cookie comme une chaîne-set-cookie (définie ci-dessous).

Note : l'algorithme ci-dessous est plus permissif que la grammaire du paragraphe 4.1. Par exemple, l'algorithme supprime les espaces en tête et en queue du nom et de la valeur du cookie (mais conserve une espace interne) tandis que la grammaire du paragraphe 4.1 interdit les espaces dans ces positions. Les agents d'utilisateur se servent de cet algorithme afin d'interopérer avec les serveurs qui ne suivent pas les recommandations de la Section 4.

Un agent d'utilisateur DOIT utiliser un algorithme équivalent à l'algorithme suivant pour analyser une "chaîne-set-cookie" :

1. Si la chaîne-set-cookie contient un caractère %x3B (";") :
   la chaîne paire-nom-valeur consiste en les caractères jusqu'à, non inclus, le premier %x3B (";"), et les attributs non analysés consistent en le reste de la chaîne-set-cookie (incluant le %x3B (";") en question).

Autrement :

   la chaîne paire-nom-valeur consiste en tous les caractères contenus dans la chaîne-set-cookie, et les attributs non analysés sont la chaîne vide.

2. Si la chaîne paire-nom-valeur n'a pas de caractère %x3D ("="), ignorer entièrement la chaîne-set-cookie.
3. La chaîne (éventuellement vide) nom consiste en les caractères jusqu'à, non inclus, le premier caractère %x3D ("=") et la chaîne valeur (éventuellement vide) consiste en les caractères après le premier caractère %x3D ("=").
4. Retirer tous les caractères WSP en tête ou en queue de la chaîne de nom et de la chaîne de valeur
5. Si la chaîne de nom est vide, ignorer entièrement la chaîne-set-cookie.
6. Le nom du cookie est la chaîne de nom, et la valeur du cookie est la chaîne valeur.

L'agent d'utilisateur DOIT utiliser un algorithme équivalent à l'algorithme suivant pour analyser les attributs non analysés :
1. Si la chaîne attributs non analysés est vide, sauter le reste de ces étapes.
2. Éliminer le premier caractère des attributs non analysés (qui sera un caractère %x3B (";")).
3. Si le reste des attributs non analysés contient un caractère %x3B (";") :
   Consommer les caractères des attributs non analysés jusqu'à, non inclus, le premier caractère %x3B (";").
   Autrement :
   Consommer le reste des attributs non analysés.

Soit la chaîne cookie-av les caractères consommés dans cette étape.

4. Si la chaîne cookie-av contient un caractère %x3D ("=") :
   La chaîne (éventuellement vide) nom-d'attribut consiste en les caractères jusqu'à, non inclus, le premier caractère %x3D ("=") et la chaîne (éventuellement vide) valeur-d'attribut consiste en les caractères après le premier caractère %x3D ("=").
   Autrement :
   La chaîne nom-d'attribut consiste en la chaîne cookie-av entière, et la chaîne valeur-d'attribut est vide.
5. Retirer tous les caractères WSP en tête ou en queue de la chaîne nom-d'attribut et de la chaîne valeur-d'attribut.
6. Traiter le nom-d'attribut et la valeur-d'attribut conformément aux exigences des paragraphes qui suivent. (Noter que les attributs avec des noms d'attribut non reconnus sont ignorés.)
7. Retourner à l'étape 1 de cet algorithme.

Quand l'agent d'utilisateur a fini d'analyser la chaîne-set-cookie, l'agent d'utilisateur est dit avoir "reçu un cookie" provenant de l'uri-de-demande qui a le nom nom-de-cookie, la valeur valeur-de-cookie et les attributs de liste-d'attributs-de-cookie. (Voir au paragraphe 5.3 les exigences supplémentaires déclenchées par la réception d'un cookie.)

#### 5.2.1 Attribut Expires

Si le nom-d'attribut correspond sans considération de la casse à la chaîne "Expires", l'agent d'utilisateur DOIT traiter le cookie-av comme suit :

Soit heure-d'expiration le résultat de l'analyse de la valeur-d'attribut comme cookie-date (voir le paragraphe 5.1.1).
Si la valeur-d'attribut a échoué à l'analyse comme date de cookie, ignorer le cookie-av.
Si l'heure-d'expiration est plus tard que la dernière date que l'agent d'utilisateur peut représenter, l'agent d'utilisateur PEUT remplacer l'heure-d'expiration par la dernière date représentable.
Si l'heure-d'expiration est plus tôt que la première date que l'agent d'utilisateur peut représenter, l'agent d'utilisateur PEUT remplacer l'heure-d'expiration par la première date représentable.
Ajouter un attribut à la liste-d'attributs-de-cookie avec un nom-d'attribut de Expires et une valeur-d'attribut de heure-d'expiration.

#### 5.2.2 Attribut Max-Age

Si le nom-d'attribut correspond sans considération de la casse à la chaîne "Max-Age", l'agent d'utilisateur DOIT traiter le cookie-av comme suit :
Si le premier caractère de la valeur-d'attribut n'est pas un CHIFFRE ou un caractère "-", ignorer le cookie-av.
Si le reste de valeur-d'attribut contient un caractère non CHIFFRE, ignorer le cookie-av.
Soit delta-secondes la valeur-d'attribut convertie en un entier.
Si delta-secondes est inférieur ou égal à zéro (0), soit heure-d'expiration la date et heure la plus tôt représentable. Autrement, soit heure-d'expiration la date et heure courante plus delta-secondes secondes.
Ajouter un attribut à la liste-d'attributs-de-cookie avec un nom-d'attribut de Max-Age et une valeur-d'attribut de heure-d'expiration.

#### 5.2.3. Attribut Domain

Si le nom-d'attribut correspond sans considération de la casse à la chaîne "Domain", l'agent d'utilisateur DOIT traiter le

cookie-av comme suit :

Si valeur-d'attribut est vide, le comportement est indéfini. Cependant, l'agent d'utilisateur DEVRAIT ignorer les cookie-av entièrement.
Si le premier caractère de la chaîne valeur-d'attribut est %x2E (".") :
Soit cookie-domaine la valeur-d'attribut sans le caractère %x2E (".") en tête.
Autrement :
Soit cookie-domaine la valeur-d'attribut entière.
Convertir le cookie-domaine en minuscules.
Ajouter un attribut à la liste-d'attributs-de-cookie avec un nom-d'attribut de Domaine et une valeur-d'attribut de cookie-domaine.

#### 5.2.4 Attribut Path

Si le nom-d'attribut correspond sans considération de la casse à la chaîne "Path", l'agent d'utilisateur DOIT traiter le cookie-av comme suit :

Si la valeur-d'attribut est vide ou si le premier caractère de valeur-d'attribut n'est pas %x2F ("/") :
Soit cookie-path le chemin par défaut.
Autrement :
Soit cookie-path la valeur-d'attribut.
Ajouter un attribut à la liste-d'attributs-de-cookie avec un nom d'attribut de Path et une valeur-d'attribut de cookie-path.

#### 5.2.5 Attribut Secure

Si le nom-d'attribut correspond sans considération de la casse à la chaîne "Secure", l'agent d'utilisateur DOIT ajouter un attribut à la liste-d'attributs-de-cookie avec un nom-d'attribut de Secure et une valeur-d'attribut vide.

#### 5.2.6 Attribut HttpOnly

Si le nom-d'attribut correspond sans considération de la casse à la chaîne "HttpOnly", l'agent d'utilisateur DOIT ajouter un attribut à la liste-d'attributs-de-cookie avec un nom-d'attribut de HttpOnly et une valeur-d'attribut vide.

### 5.3 Modèle de mémorisation

L'agent d'utilisateur mémorise les champs suivants sur chaque cookie : nom, valeur, heure-d'expiration, domaine, chemin, heure -de-création, dernière-heure-d'accès, fanion-persistant, fanion-seulement-hôte, fanion-secure-seulement, et http-seulement.

Quand l'agent d'utilisateur "reçoit un cookie" provenant d'un uri-de-demande avec un nom nom-de-cookie, une valeur valeur-de-cookie, et des attributs liste-d'attributs-de-cookie, l'agent d'utilisateur DOIT traiter le cookie comme suit :

1. Un agent d'utilisateur PEUT ignorer en totalité un cookie reçu. Par exemple, l'agent d'utilisateur peut souhaiter bloquer la réception des cookies provenant de réponses de "tiers" ou l'agent d'utilisateur peut ne pas souhaiter mémoriser les cookies qui excèdent une certaine taille.
2. Créer un nouveau cookie avec le nom nom-de-cookie, la valeur valeur-de-cookie. Régler l'heure de création et l'heure de dernier accès à la date et heure actuelles.
3. Si la liste-d'attributs-de-cookie contient un attribut avec un nom-d'attribut de "Max-Age":
   Régler le fanion persistant du cookie à vrai.
   Régler l'heure d'expiration du cookie à la valeur-d'attribut du dernier attribut de la liste-d'attributs-de-cookie avec un nom-d'attribut de "Max-Age".
   Autrement, si la liste-d'attributs-de-cookie contient un attribut avec un nom-d'attribut de "Expires" (et ne contient pas d'attribut avec un nom-d'attribut de "Max-Age"):
   Régler le fanion persistant du cookie à vrai.
   Régler l'heure d'expiration du cookie à la valeur-d'attribut du dernier attribut de la liste-d'attributs-de-cookie avec un nom-d'attribut de "Expires".
   Autrement :
   Régler le fanion persistant du cookie à faux .
   Régler l'heure d'expiration du cookie à la date la plus tardive représentable.
4. Si la liste-d'attributs-de-cookie contient un attribut avec un nom-d'attribut de "Domain" :

Soit attribut-de-domaine la valeur-d'attribut du dernier attribut de la liste-d'attributs-de-cookie avec un nom-d'attribut de "Domain".
Autrement :
Soit attribut-de-domaine la chaîne vide.

5. Si l'agent d'utilisateur est configuré à rejeter les "suffixes publics" et si attribut-de-domaine est un suffixe public :
Si l'attribut-de-domaine est identique à l'hôte de demande canonisé,
Soit attribut-de-domaine la chaîne vide.
Autrement :
Ignorer le cookie dans sa totalité et interrompre ces étapes.

Note : un "suffixe public" est un domaine contrôlé par un registre public, tel que "com", "co.uk", et "pvt.k12.wy.us". Cette étape est essentielle pour empêcher un attaquant.com de perturber l'intégrité de exemple.com en établissant un cookie avec un attribut Domain de "com". Malheureusement, l'ensemble des suffixes publics (aussi appelé "domaines contrôlés par un registre") change au fil du temps. Si c'est faisable, les agents d'utilisateur DEVRAIENT utiliser une liste à jour des suffixes publics, comme celle tenue par le projet Mozilla à < http://publicsuffix.org/ >.

6. Si attribut-de-domaine est non vide :
Si l'hôte-de-demande canonisé ne correspond pas au nom de domaine de l'attribut-de-domaine :
Ignorer entièrement le cookie et interrompre ces étapes.
Autrement :
Régler le fanion seulement-hôte du cookie à faux.
Régler le domaine du cookie à attribut-de-domaine.
Autrement :
Régler le fanion seulement-hôte du cookie à vrai.
Régler le domaine du cookie à hôte-de-demande canonisé.

7. Si la liste-d'attributs-de-cookie contient un attribut avec un nom-d'attribut de "Path", régler le chemin du cookie à la valeur-d'attribut du dernier attribut de la liste-d'attributs-de-cookie avec un nom-d'attribut de "Path". Autrement, régler le chemin du cookie au chemin par défaut de l'uri-de-demande.

8. Si la liste-d'attributs-de-cookie contient un attribut avec un nom-d'attribut de "Secure", régler le fanion secure-seulement du cookie à vrai. Autrement, régler le fanion secure-seulement du cookie à faux.

9. Si la liste-d'attributs-de-cookie contient un attribut avec un nom-d'attribut de "HttpOnly", régler le fanion seulement-http du cookie à vrai. Autrement, régler le fanion seulement-http du cookie à faux.

10. Si le cookie a été reçu d'une API "non HTTP" et si le fanion seulement-http du cookie est établi, interrompre ces étapes et ignorer entièrement le cookie.

11. Si le magasin de cookies contient un cookie avec le même nom, domaine, et chemin que le cookie nouvellement créée :
    1. Soit vieux-cookie le cookie existant avec le même nom, domaine, et chemin que le cookie nouvellement créé. (Noter que cet algorithme conserve l'invariant qu'il y a au plus un tel cookie.)
    2. Si le cookie nouvellement créé a été reçu d'une API "non HTTP" et si le fanion seulement-http du vieux cookie est établi, interrompre ces étapes et ignorer entièrement le cookie nouvellement créé.
    3. Mettre à jour l'heure-de-création du cookie nouvellement créé pour qu'elle corresponde à l'heure-de-création du vieux cookie.
    4. Retirer le vieux cookie du magasin de cookies.

12. Insérer le cookie nouvellement créé dans le magasin de cookies.

Un cookie est "expiré" si il a une date d'expiration qui est passée.

L'agent d'utilisateur DOIT évincer tous les cookies expirés du magasin de cookies si, à tout moment, il existe un cookie expiré dans le magasin de cookies.

À tout moment, l'agent d'utilisateur PEUT "supprimer des cookies en excès" du magasin de cookies si le nombre de cookies partageant un champ de domaine excède un seuil défini par la mise en œuvre (comme 50 cookies).

À tout moment, l'agent d'utilisateur PEUT "supprimer des cookies en excès" du magasin de cookies si le magasin de cookies excède un seuil prédéterminé (par exemple, 3000 cookies).

Quand l'agent d'utilisateur supprime un excès de cookies du magasin de cookies, il DOIT évincer les cookies dans l'ordre de priorité suivant :
1. cookies arrivés à expiration
2. cookies qui partagent un champ de domaine avec plus qu'un nombre prédéterminé d'autres cookies
3. tous les cookies.

Si deux cookies ont la même priorité de suppression, l'agent d'utilisateur DOIT évincer d'abord le cookie avec la plus ancienne date-de-dernier-accès.

Quand "la session en cours est terminée" (comme défini par l'agent d'utilisateur) l'agent d'utilisateur DOIT supprimer du magasin de cookies tous les cookies dont le fanion persistant est réglé à faux.

### 5.4 En-tête Cookie

L'agent d'utilisateur inclut les cookies mémorisés dans l'en-tête Cookie de demande HTTP.

Quand l'agent d'utilisateur génère une demande HTTP, l'agent d'utilisateur NE DOIT PAS rattacher plus d'un champ d'en-tête Cookie.

Un agent d'utilisateur PEUT omettre entièrement l'en-tête Cookie. Par exemple, l'agent d'utilisateur peut souhaiter bloquer l'envoi de cookies durant des demandes de "tiers" pour établir des cookies (voir le paragraphe 7.1).

Si l'agent d'utilisateur joint un champ d'en-tête Cookie à une demande HTTP, l'agent d'utilisateur DOIT envoyer la chaîne-de-cookie (définie ci-dessous) comme la valeur du champ d'en-tête.

L'agent d'utilisateur DOIT utiliser un algorithme équivalent à l'algorithme suivant pour calculer la "chaîne-de-cookie" à partir d'un magasin de cookies et d'un uri-de-demande :

1.Soit cookie-liste l'ensemble des cookies du magasin de cookies qui satisfont à toutes les exigences suivantes :
   * soit : le fanion seulement-hôte du cookie est vrai et l'hôte-de-demande canonisé est identique au domaine du cookie.
     soit : le fanion seulement-hôte du cookie est faux et l'hôte-de-demande canonisé correspond en domaine à celui du cookie.
   * Le chemin de l'uri-de-demande correspond en chemin à celui du cookie.
   * Si le fanion seulement-sûr du cookie est vrai, alors le schéma d'uri-de-demande doit noter un protocole "sûr" (comme défini par l'agent d'utilisateur).
     Note : la notion de protocole "sûr" n'est pas définie par le présent document. Normalement, les agents d'utilisateur considèrent qu'un protocole est sûr si il fait usage de la sécurité de la couche transport, comme SSL ou TLS. Par exemple, la plupart des agents d'utilisateur considèrent "https" comme un schéma de protocole sûr.
   * Si le fanion seulement-http du cookie est vrai, exclure alors le cookie si la chaîne-de-cookie est générée pour une API "non HTTP" (comme défini par l'agent d'utilisateur).

2. L'agent d'utilisateur DEVRAIT trier la liste de cookies dans l'ordre suivant :
   * les cookies avec les chemins les plus longs avant ceux qui ont des chemins plus courts.
   * parmi les cookies qui ont des champs de chemin d'égale longueur, les cookies avec une date de création antérieure sont avant ceux qui ont des dates de création postérieures.
     Note : tous les agents d'utilisateur ne trient pas la liste de cookies dans cet ordre, mais cet ordre reflète la pratique courante au moment de la rédaction, et historiquement, il y a eu des serveurs qui (à tort) dépendaient de cet ordre.

3. Mettre à jour l'heure-du-dernier-accès de chaque cookie dans la liste-des-cookies à la date et heure actuelles.

4. Mettre à la suite la liste-de-cookies dans une chaîne-de-cookie en traitant chaque cookie de la liste dans l'ordre :
   1. Sortir le nom de cookie, le caractère %x3D ("=") et la valeur du cookie.
   2. Si il y a un cookie non traité dans la liste-de-cookies, sortir les caractères %x3B et %x20 ("; ").
      Note : en dépit de son nom, la chaîne-de-cookie est en fait une séquence d'octets, non une séquence de caractères. Pour convertir la chaîne-de-cookie (ou ses composants) en une séquence de caractères (par exemple, pour la présentation à l'utilisateur) l'agent d'utilisateur peut souhaiter essayer d'utiliser le codage de caractères UTF-8 [RFC3629] pour décoder la séquence d'octets. Ce décodage peut cependant échouer parce que toutes les séquence d'octets ne sont pas de l'UTF-8 valide.

# 6. Considérations de mise en œuvre

## 6.1 Limites

Les mises en œuvre pratiques d'agent d'utilisateur ont des limites quant au nombre et à la taille des cookies qu'elles peuvent mémoriser. Les agents d'utilisateur d'usage général DEVRAIENT fournir les capacités minimales suivantes :

- Au moins 4096 octets par cookie (mesuré par la somme des longueurs du nom, de la valeur et des attributs du cookie).
- Au moins 50 cookies par domaine.
- Au moins 3000 cookies au total.

Les serveurs DEVRAIENT utiliser aussi peu de cookies que possible et aussi courts que possible pour éviter d'atteindre les limites de ces mises en œuvre et minimiser la consommation de bande passante du réseau due à l'inclusion de l'en-tête Cookie dans chaque demande.

Les serveurs DEVRAIENT rétrograder en douceur si l'agent d'utilisateur manque à retourner un ou plusieurs cookies dans l'en-tête Cookie parce que l'agent d'utilisateur pourrait évincer tout cookie à tout moment sur l'ordre de l'utilisateur.

## 6.2 Interfaces de programmation d'application

Une raison de l'utilisation d'une syntaxe aussi ésotérique par les en-têtes Cookie et Set-Cookie est que de nombreuses plateformes (à la fois chez les serveurs et chez les agents d'utilisateur) fournissent des API fondées sur la chaîne aux cookies, exigeant des programmeurs de couche d'application qu'ils génèrent et analysent la syntaxe utilisée par les en-têtes Cookie et Set-Cookie, ce que beaucoup ont fait de façon incorrecte, résultant en problèmes d'interopérabilité.

Au lieu de fournir aux cookies des API fondées sur la chaîne, les plateformes seraient mieux servies en fournissant plus de sémantique aux API. Il sort du domaine d'application du présent document de recommander des conceptions d'API spécifiques, mais il y a de clairs avantages à accepter un objet "Date"' abstrait au lieu d'une chaîne de date mise en série.

## 6.3 Dépendance à IDNA et migration

IDNA2008 [RFC5890] se substitue à IDNA2003 [RFC3490]. Cependant, il y a des différences entre les deux spécifications, et donc il peut y avoir des différences dans le traitement (par exemple, la conversion) des étiquettes de nom de domaine qui ont été enregistrées sous l'une à partir de celles enregistrées sous l'autre. Il y aura une période de transition durant laquelle les étiquettes de nom de domaines fondées sur IDNA2003 vont exister dans la nature. Les agents d'utilisateur DEVRAIENT mettre en œuvre IDNA2008 [RFC5890] et PEUVENT mettre en œuvre [UTS46] ou [RFC5895] afin de faciliter la transition IDNA. Si un agent d'utilisateur ne met pas en œuvre IDNA2008, il DOIT mettre en œuvre IDNA2003 [RFC3490].

# 7. Considérations de confidentialité

Les cookies sont souvent critiqués pour laisser les serveurs tracer les usagers. Par exemple, un certain nombre de sociétés "d'analyse de la Toile" utilisent des cookies pour reconnaître quand un utilisateur retourne sur un site de la Toile ou visite un autre site. Bien que les cookies ne soient pas le seul mécanisme que les serveurs peuvent utiliser pour tracer les utilisateurs à travers les demandes HTTP, les cookies facilitent le traçage parce que ils sont persistants à travers les sessions d'agent d'utilisateur et peuvent être partagés entre des hôtes.

## 7.1 Cookies de tiers

Les cookies dits "de tiers" causent des soucis particuliers. En rendant un document HTML, un agent d'utilisateur demande souvent des ressources à d'autres serveurs (comme des réseaux d'annonces). Ces serveurs tiers peuvent utiliser des cookies pour tracer l'utilisateur même si il n'a jamais visité directement le serveur. Par exemple, si un utilisateur visite un site qui a des contenus qui proviennent d'un tiers et ensuite visite un autre site qui a des contenus provenant du même tiers, le tiers peut tracer l'utilisateur entre les deux sites.

Certains agents d'utilisateur restreignent la façon dont les cookies de tiers se comportent. Par exemple, certains de ces agents d'utilisateur refusent d'envoyer l'en-tête Cookie dans des demandes de tiers. D'autres refusent de traiter les en-tête Set-Cookie dans des réponses à des demandes de tiers. Les agents d'utilisateur ont de grandes différences de politique à l'égard des cookies de tiers. Le présent document accorde aux agents d’utilisateur une grande latitude pour faire des expériences avec des politiques de cookie de tiers qui tiennent un équilibre entre les besoins de confidentialité et de compatibilité de leurs utilisateurs. Cependant, le présent document ne préconise aucune politique particulière de cookie de

tiers.

Les politiques de blocage des cookies de tiers sont souvent inefficaces pour réaliser des objectifs de confidentialité si les serveurs tentent de contourner les interdictions de traçage des utilisateurs. En particulier, deux serveurs qui collaborent peuvent souvent tracer les usagers sans utiliser du tout de cookies en injectant des informations d'identification dans des URL dynamiques.

### 7.2 Commandes d'utilisateur

Les agents d'utilisateur DEVRAIENT fournir aux usagers un mécanisme pour gérer les cookies mémorisés dans les magasins de cookies. Par exemple, un agent d'utilisateur peut laisser les usagers supprimer tous les cookies reçus durant une période spécifiée ou tous les cookies relatifs à un domaine particulier. De plus, de nombreux agents d'utilisateur incluent un élément d'interface d'utilisateur pour permettre aux usagers d'examiner les cookies mémorisés dans leur magasin de cookies.

Les agents d'utilisateur DEVRAIENT fournir aux usagers un mécanisme pour désactiver les cookies. Quand les cookies sont désactivés, l'agent d'utilisateur NE DOIT PAS inclure un en-tête Cookie dans les demandes HTTP sortantes et l'agent d'utilisateur NE DOIT PAS traiter les en-têtes Set-Cookie dans les réponses HTTP entrantes.

Certains agents d'utilisateur fournissent aux usagers l'option d'empêcher la mémorisation persistante des cookies à travers les sessions. Quand il est configuré de façon fiable, l'agent d'utilisateur DOIT traiter tous les cookies reçus comme si le fanion persistant était réglé à faux. Certains agents d'utilisateur courants exposent cette fonctionnalité via le mode "navigation confidentielle" [Aggarwal2010].

Certains agents d'utilisateur fournissent aux usagers la capacité d'approuver les écritures individuelles dans le magasin de cookies. Dans de nombreux scénarios d'usage courants, ces commandes génèrent un grand nombre d'invites. Cependant, des usagers conscients de la nécessité de préserver leur intimité trouvent que ces commandes sont néanmoins utiles.

### 7.3 Dates d'expiration

Bien que les serveurs puissent régler la date d'expiration des cookies à un futur lointain, la plupart des agents d'utilisateur ne conservent pas en fait les cookies pendant plusieurs dizaines de jours. Plutôt que de choisir des périodes d'expiration gratuitement longues, les serveurs DEVRAIENT promouvoir l'intimité de l'utilisateur en choisissant des périodes raisonnables d'expiration de cookies sur la base de l'objet du cookie. Par exemple, un identifiant de session normal pourrait raisonnablement être réglé à expirer en deux semaines.

## 8. Considérations sur la sécurité

### 8.1 Généralités

Les cookies posent un certain nombre de défis à la sécurité. Cette section couvre quelques un des problèmes les plus saillants.

En particulier, les cookies encouragent les développeurs à s'appuyer sur l'autorité ambiante pour l'authentification, devenant souvent vulnérables aux attaques comme la falsification de demande trans-site [CSRF]. Aussi, quand ils mémorisent les identifiants de session dans des cookies, les développeurs créent souvent des vulnérabilités de fixation de session.

Le chiffrement de la couche transport, comme celui employé dans HTTPS, est insuffisant pour empêcher un attaquant du réseau d'obtenir ou d'altérer les cookies d'une victime parce que le protocole de cookie lui-même a diverses vulnérabilités (voir "Confidentialité faible" et "Intégrité faible", ci-dessous). De plus, par défaut, les cookies n'assurent pas la confidentialité ou l'intégrité contre les attaquants du réseau, même quand ils sont utilisés en conjonction avec HTTPS.

### 8.2 Autorité ambiante

Un serveur qui utilise des cookies pour authentifier les utilisateurs peut accepter des vulnérabilités de sécurité parce que certains agents d'utilisateur laissent des parties distantes produire des demandes HTTP à partir de l'agent d'utilisateur (par exemple, via des redirections HTTP ou des formulaires HTML). Quand ils produisent ces demandes, les agents d'utilisateur attachent des cookies même si la partie distante ne connaît pas le contenu des cookies, laissant potentiellement la partie distante exercer l'autorité d'un serveur imprudent.

Bien que ces problèmes de sécurité passent par un certain nombre de noms (par exemple, falsification de demande trans-

site, mandataire confus) le problème découle des cookies comme forme d'autorité ambiante. Les cookies encouragent les opérateurs de serveur à séparer la désignation (sous la forme des URL) de l'autorisation (sous la forme de cookies). Par conséquent, l'agent d'utilisateur peut fournir l'autorisation pour une ressource désignée par l'attaquant, causant éventuellement l'entreprise par le serveur ou ses clients d'actions désignées par l'attaquant bien qu'elles aient été autorisées par l'usager.

Au lieu d'utiliser des cookies pour l'autorisation, les opérateurs de serveur pourraient souhaiter entremêler la désignation et l'autorisation en traitant les URL comme des capacités. Au lieu de mémoriser les secrets dans les cookies, cette approche mémorise les secrets dans les URL, ce qui exige que l'entité distante fournisse le secret lui-même. Bien que cette approche ne soit pas une panacée, une application judicieuse de ces principes peut conduire à une sécurité plus robuste.

### 8.3 Texte en clair

Sauf si elles sont envoyées sur un canal sûr (comme TLS) les information des en-têtes Cookie Set-Cookie sont transmises en clair.

1. Toutes les informations sensibles convoyées dans ces en-têtes sont exposées à l'espionnage.
2. Un intermédiaire malveillant pourrait altérer les en-têtes lorsque ils voyagent dans l'une ou l'autre direction, avec des résultats imprévisibles.
3 Un client malveillant pourrait altérer l'en-tête Cookie avant la transmission, avec des résultats imprévisibles.

Les serveurs DEVRAIENT chiffrer et signer le contenu des cookies (en utilisant tout format que le serveur désire) quand ils les transmettent à l'agent d'utilisateur (même quand les cookies sont envoyés sur un canal sûr). Cependant, chiffrer et signer le contenu du cookie n'empêche pas un attaquant de transplanter un cookie d'un agent d'utilisateur à un autre ou de répéter le cookie ultérieurement.

En plus du chiffrement et de la signature du contenu de chaque cookie, les serveurs qui requièrent un niveau de sécurité supérieur DEVRAIENT utiliser les en-têtes Cookie et Set-Cookie seulement sur un canal sûr. Quand ils utilisent des cookies sur un canal sûr, les serveurs DEVRAIENT établir l'attribut Secure (voir au paragraphe 4.1.2.5) pour chaque cookie. Si un serveur n'établit pas l'attribut Secure, la protection fournie par le canal sûr sera largement illusoire.

Par exemple, considérons un serveur de messagerie qui mémorise un identifiant de session dans un cookie et auquel on accède normalement sur HTTPS. Si le serveur n'établit pas l'attribut Secure sur ses cookies, un attaquant actif du réseau peut intercepter toute demande HTTP sortante de l'agent d'utilisateur et rediriger cette demande au serveur de messagerie sur HTTP. Même si le serveur de messagerie n'écoute pas les connexions HTTP, l'agent d'utilisateur va quand même inclure les cookies dans la demande. L'attaquant actif du réseau peut intercepter ces cookies, les répéter contre le serveur, et découvrir le contenu de la messagerie de l'usager. Si le serveur avait plutôt établi l'attribut Secure sur ses cookies, l'agent d'utilisateur n'aurait pas inclus de cookies dans sa demande en clair.

### 8.4 Identifiants de session

Au lieu de mémoriser les informations de session directement dans un cookie (où il pourrait être exposé à un attaquant ou répété par lui) les serveurs mémorisent habituellement un nom occasionnel (ou "identifiant de session") dans un cookie. Quand le serveur reçoit une demande HTTP avec un nom occasionnel, il peut chercher les informations d'état associées au cookie en utilisant le nom occasionnel comme clé.

Utiliser des cookies comme identifiant de session limite les dommages qu'un attaquant peut causer si il append le contenu d'un cookie parce que le nom occasionnel n'est utile que pour interagir avec le serveur (à la différence d'un contenu de cookie qui n'est pas un nom occasionnel, qui pourrait être lui-même sensible). De plus, utiliser un seul nom occasionnel empêche un attaquant de "lier" ensemble le contenu des cookies provenant de deux interactions avec le serveur, ce qui pourrait causer un comportement inattendu du serveur.

Utiliser des identifiants de session n'est pas sans risque. Par exemple, le serveur DEVRAIT prendre soin d'éviter les vulnérabilités de "fixation de session". Une attaque de fixation de session procède en trois étapes. D'abord, l'attaquant transplante un identifiant de session de son agent d'utilisateur à celui de la victime. Ensuite, la victime utilise cet identifiant de session pour interagir avec le serveur, éventuellement en ajoutant à l'identifiant de session des accréditifs ou des informations confidentielles de l'utilisateur. Enfin, l'attaquant utilise l'identifiant de session pour interagir directement avec le serveur, obtenant éventuellement les informations d'autorité ou confidentielles de l'utilisateur.

### 8.5 Confidentialité faible

Les cookies n'assurent pas l'isolation par accès. Si un cookie est lisible par un service qui fonctionne sur un accès, le cookie est aussi lisible par un service qui fonctionne sur un autre accès du même serveur. Si un cookie est en écriture pour un service sur un accès, il l'est aussi par un service qui fonctionne sur un autre accès du même serveur. Pour cette raison, les serveurs NE DEVRAIENT PAS faire fonctionner des services qui ne sont pas mutuellement de confiance sur des accès différents du même hôte et utiliser des cookies pour mémoriser des informations sensibles.

Les cookies n'assurent pas l'isolation par schéma. Bien que très couramment utilisés avec les schémas http et https, les cookies pour un certain hôte peuvent aussi être disponibles aux autres schémas, tels que ftp et gopher. Bien que ce manque d'isolation par schéma soit le plus apparent dans les API non HTTP qui permettent l'accès aux cookies (par exemple, l'API document.cookie de HTML) le manque d'isolation par schéma est en fait présent dans les exigences pour le traitement des cookies eux-mêmes (par exemple, considérer la restitution d'un URI avec le schéma gopher via HTTP).

Les cookies ne fournissent pas toujours l'isolation par chemin. Bien que le protocole de niveau réseau n'envoie pas les cookies mémorisés pour un chemin à un autre, certains agents d'utilisateur exposent les cookies via des API non HTTP, comme l'API document.cookie de HTML Parce que certains de ces agents d'utilisateur (par exemple, les navigateurs de la Toile) n'isolent pas les ressources reçues de différents chemins, une ressource restituée d'un chemin pourrait être capable d'accéder aux cookies mémorisés pour un autre chemin.

### 8.6. Intégrité faible

Les cookies ne fournissent pas de garantie d'intégrité pour les domaines parents (et leurs sous domaines). Par exemple, considérons foo.exemple.com et bar.exemple.com. Le serveur foo.exemple.com peut établir un cookie avec un attribut Domaine de "exemple.com" (éventuellement écrasant un cookie "exemple.com" existant établi par bar.exemple.com) et l'agent d'utilisateur va inclure ce cookie dans des demandes HTTP à bar.exemple.com. Dans le pire des cas, bar.exemple.com sera incapable de distinguer ce cookie d'un cookie qu'il a lui-même établi. Le serveur foo.exemple.com pourrait être capable de se servir de cette capacité pour monter une attaque contre bar.exemple.com.

Bien que l'en-tête Set-Cookie accepte l'attribut Path, l'attribut Path ne fournit aucune protection de l'intégrité parce que l'agent d'utilisateur va accepter un attribut Path arbitraire dans un en-tête Set-Cookie. Par exemple, une réponse HTTP à une demande pour http://exemple.com/foo/bar peut établir un cookie avec un attribut Path de "/qux". Par conséquent, les serveurs NE DEVRAIENT PAS faire fonctionner à la fois des services qui ne sont pas mutuellement de confiance sur des chemins différents du même hôte et utiliser des cookies pour mémoriser des informations sensibles de sécurité.

Un attaquant actif du réseau peut aussi injecter des cookies dans l'en-tête Cookie envoyé à https://exemple.com/ en se faisant passer pour une réponse provenant de http://exemple.com/ et en injectant un en-tête Set-Cookie. Le serveur HTTPS à exemple.com sera incapable de distinguer ces cookies de ceux qui a lui-même établi dans une réponse HTTPS. Un attaquant actif du réseau pourrait être capable de se servir de cette capacité pour monter une attaque contre exemple.com même si exemple.com utilise exclusivement HTTPS.

Les serveurs peuvent partiellement atténuer ces attaques en chiffrant et signant le contenu de leurs cookies. Cependant, utiliser la cryptographie n'atténue pas complètement le problème parce que un attaquant peut répéter un cookie qu'il a reçu de l'authentique serveur exemple.com dans la session de l'utilisateur, avec des résultats imprévisibles.

Finalement, un attaquant pourrait être capable de forcer l'agent d'utilisateur à supprimer des cookies en mémorisant un grand nombre de cookies. Une fois que l'agent d'utilisateur a atteint sa limite de mémorisation, l'agent d'utilisateur sera forcé d'évincer certains cookies. Les serveurs NE DEVRAIT PAS se fier à ce que les agents d'utilisateur vont conserver les cookies.

### 8.7 Fiabilité du DNS

Les cookies s'appuient sur le système des noms de domaines (DNS) pour la sécurité. Si le DNS est partiellement ou complètement compromis, le protocole de cookie peut échouer à fournir les propriétés de sécurité exigées par les applications.

## 9. Considérations relatives à l'IANA

Le registre des champ d'en-tête permanents de messages (voir la [RFC3864]) a été mis à jour avec les enregistrements suivants.

### 9.1 Cookie

Nom de champ d'en-tête : Cookie
Protocole applicable : http
Statut : standard
Auteur/Contrôleur des changements : IETF
Document de spécification : RFC6265, paragraphe 5.4

### 9.2 Set-Cookie

Nom de champ d'en-tête : Set-Cookie
Protocole applicable : http
Statut : standard
Auteur/Contrôleur des changements : IETF
Document de spécification : RFC6265, paragraphe 5.2)

### 9.3 Cookie2

Nom de champ d'en-tête : Cookie2
Protocole applicable : http
Statut : obsolète
Auteur/Contrôleur des changements : IETF
Document de spécification : [RFC2965]

### 9.4 Set-Cookie2

Nom de champ d'en-tête : Set-Cookie2
Protocole applicable : http
Statut : obsolète
Auteur/Contrôleur des changements : IETF
Document de spécification : [RFC2965]

## 10. Références

### 10.1 Références normatives

[RFC1034] P. Mockapetris, "Noms de domaines - Concepts et facilités", STD 13, novembre 1987. (*MàJ par* RFC1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936, 8020, 8482, 8767)

[RFC1123] R. Braden, éditeur, "Exigences pour les hôtes Internet – Application et prise en charge", STD 3, octobre 1989. *(MàJ par* RFC7766)

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC2616] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", juin 1999. *(D.S., MàJ par* 2817, 6585)

[RFC3490] P. Faltstrom et autres, "Internationalisation des noms de domaine dans les applications (IDNA)", mars 2003. (*Remplacée par les* RFC5890 et 5891, *P.S.*) Voir au paragraphe 6.3 l'explication de pourquoi est nécessaire une référence normative à une spécification obsolète.

[RFC4790] C. Newman et autres, "Registre de collationnement des protocoles d'application de l'Internet", mars 2007. *(P.S.)*

[RFC5234] D. Crocker, éd., P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", janvier 2008. (STD0068)

[RFC5890] J. Klensin, "Noms de domaine internationalisés pour les applications (IDNA) : Définitions et cadre documentaire", août 2010. *(Remplace RFC*3490) *(P.S.)*

[USASCII] American National Standards Institute, "Coded Character Set -- 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.

### 10.2 Références pour information

[Aggarwal2010] Aggarwal, G., Burzstein, E., Jackson, C., et D. Boneh, "An Analysis of Private Browsing Modes in Modern Browsers", 2010, < http://www.usenix.org/events/sec10/tech/full_papers/Aggarwal.pdf >.

[CSRF] Barth, A., Jackson, C., et J. Mitchell, "Robust Defenses for Cross-Site Request Forgery", 2008, <http://portal.acm.org/citation.cfm?id=1455770.1455782 >.

[Kri2001] Kristol, D., "HTTP Cookies: Standards, Privacy, et Politics", ACM Transactions on Internet Technology Vol. 1, #2, novembre 2001, < http://arxiv.org/abs/cs.SE/0105018 >.

[Netscape] Netscape Communications Corp., "Persistent Client State -- HTTP Cookies", 1999, <http://web.archive.org/web/20020803110822/http://wp.netscape.com/newsref/std/cookie_spec.html>.

[RFC2109] D. Kristol, L. Montulli, "Mécanisme de gestion d'état HTTP", février 1997. (*Obsolète, voir* RFC2965) *(P.S.)*

[RFC2818] E. Rescorla, "HTTP sur TLS", mai 2000. *(Information)*

[RFC2965] D. Kristol, L. Montulli, "Mécanisme de gestion d'état HTTP", octobre 2000. *(P.S.)*

[RFC3629] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", STD 63, novembre 2003.

[RFC3864] G. Klyne, M. Nottingham, J. Mogul, "Procédures d'enregistrement pour les champs d'en-tête de message", septembre 2004. (BCP0090)

[RFC4648] S. Josefsson, "Codages de données Base16, Base32 et Base64", octobre 2006. (*Remplace* RFC3548) *(P.S.)*

[RFC5895] P. Resnick, P. Hoffman, "Transposition de caractères pour les noms de domaine internationalisés pour les applications (IDNA)", septembre 2010. *(Information)*

[UTS46] Davis, M. et M. Suignard, "Unicode IDNA Compatibility Processing", Unicode Technical Standards # 46, 2010, <http://unicode.org/reports/tr46/>.

## Appendice A. Remerciements

Le présent document fait de larges emprunts à la [RFC2109]. Ce sont les efforts de David M. Kristol et Lou Montulli qui ont permis de spécifier les cookies. En particulier, David M. Kristol, af ourni des avis précieux sur le processus de navigation de l'IETF. Merci aussi à Thomas Broyer, Tyler Close, Alissa Cooper, Bil Corry, corvid, Lisa Dusseault, Roy T. Fielding, Blake Frantz, Anne van Kesteren, Eran Hammer-Lahav, Jeff Hodges, Bjoern Hoehrmann, Achim Hoffmann, Georg Koppen, Dean McNamee, Alexey Melnikov, Mark Miller, Mark Pauley, Yngve N. Pettersen, Julian Reschke, Peter Saint-Andre, Mark Seaborn, Maciej Stachowiak, Daniel Stenberg, Tatsuhiro Tsujikawa, David Wagner, Dan Winship, et Dan Witte de leurs précieux retours sur le document.

## Adresse de l'auteur

Adam Barth
University of California, Berkeley
mél : abarth@eecs.berkeley.edu
URI : http://www.adambarth.com/