Network Working Group
Request for Comments: 3986
STD: 66
Mise à jour de : 1738
Rendus obsolètes : 2732, 2396, 1808
Catégorie : Standards Track

T. Berners-Lee
W3C/MIT
R. Fielding
Day Software
L. Masinter
Adobe Systems
Janvier 2005

Identifiant de ressource uniforme (URI) : Syntaxe générique

Statut de ce mémo

Le présent document spécifie un protocole de suivi de norme Internet pour la communauté Internet, et appelle à la discussion pour formuler des suggestions d'améliorations. Se référer à l'édition en cours des normes officielles de protocole Internet "Internet Official Protocol Standards" (STD 1) pour voir l'état de la normalisation et le statut de ce protocole. La distribution de ce mémo n'est pas limitée.

Droits de propriété intellectuelle

Résumé

Un identifiant de ressources uniforme (URI, *Uniform Ressource Identifier*) est une séquence compacte de caractères qui identifient une ressource abstraite ou physique. La présente spécification définit la syntaxe générique de l'URI et un processus de résolution des références d'URI qui peut être en forme relative, ainsi que des lignes directrices et des considérations de sécurité pour l'utilisation des URI sur l'Internet. La syntaxe URI définit une grammaire qui est un sur-ensemble des tous les URI valides, qui permet à une mise en œuvre d'analyser les composants communs d'une référence d'URI sans connaître les exigences spécifiques du schéma de tout identifiant possible. La présente spécification ne définit pas une grammaire génératrice des URI ; cette tâche est effectuée par les spécifications individuelles de chaque schéma d'URI.

Table des Matières

1. Introduction

Un identifiant de ressource uniforme (URI, *Uniform Ressource Identifier*) fournit un moyen simple et extensible d'identification d'une ressource. La présente spécification de la syntaxe et de la sémantique de l'URI est déduite de concepts introduits par l'initiative d'information mondiale du World Wide Web, dont l'utilisation de ces identifiants date des années 1990 et est décrite dans "Universal Identifiants de ressource in WWW" [RFC1630]. Cette syntaxe est conçue pour satisfaire aux recommandations posées dans les "Recommandations fonctionnelles pour les localisations de ressources Internes" [RFC1736] et "Exigences fonctionnelles pour les noms de ressource uniformes" [RFC1737].

Le présent document rend obsolète la [RFC2396], qui fusionne "Uniform Ressource Locators" [RFC1738] et "Relative Uniform Ressource Locators" [RFC1808] pour définir une syntaxe générique simple pour tous les URI. Il rend obsolète la [RFC2732], qui introduisait la syntaxe pour une adresse IPv6. Il exclut les portions de RFC 1738 qui définissent la syntaxe spécifique des schémas individuels d'URI ; ces portions feront l'objet de mises à jour comme documents distincts. Le processus d'enregistrement de nouveaux schémas d'URI est défini à part par le [BCP35]. L'avis des concepteurs de nouveaux schémas d'URI se trouve dans la [RFC2718]. Tous les changements significatifs par rapport à la RFC 2396 sont notés à l'Appendice D.

La présente spécification utilise les termes "caractère" et "ensemble de caractères codés " conformément aux définitions données dans [BCP19], et "codage de caractère" à la place de ce que [BCP19] appelle un "charset" (*ensemble de caractère*).

1.1. Généralités sur les URI

Les URI se caractérisent comme suit :

Uniformes

L'uniformité procure des bénéfices variés. Elle permet d'utiliser différents types d'identifiants de ressource dans le même contexte, même lorsque les mécanismes utilisés pour accéder à ces ressources diffèrent. Elle permet une interprétation sémantique uniforme des conventions syntaxiques communes à travers les différents types d'identifiant de ressource. Elle permet l'introduction de nouveaux types d'identifiants de ressource sans s'occuper de la façon dont les identifiants existants sont utilisés. Elle permet de réutiliser les identifiants dans de nombreux contextes différents, permettant ainsi à de nouvelles applications ou protocoles de mettre à niveau un grand ensemble d'identifiants de ressource pré existant et largement utilisé.

Ressource

La présente spécification ne limite pas le domaine d'application de la notion de ressource ; le terme de "ressource" est plutôt utilisé dans un sens général pour tout ce qui peut être identifié par un URI. Les exemples les plus parlants seront ceux d'un document électronique, une image, une source d'information avec un objet approprié (par exemple, "rapport météorologique quotidien sur Los Angeles"), un service (par exemple, une passerelle HTTP-vers-SMS), et une quantité d'autres ressources. Une ressource n'est pas nécessairement accessible via l'Internet ; par exemple, êtres humains, corporations, et bound books dans une librairie peuvent aussi être des ressources. De même, des concepts abstraits peuvent être des ressources, comme les opérateurs et les opérandes d'une équation mathématique, les types de relations (par exemple, "parent" ou "employé"), ou des valeurs numériques (par exemple, zéro, un, et l'infini).

Identifiant

Un identifiant englobe les informations nécessaires pour distinguer ce qui est identifié de toutes les autres choses qui sont autour de son domaine d'identification. Notre utilisation des termes "identifie" et "identifiant" se réfère à l'objectif de distinguer une ressource de toutes les autres ressources, sans considération de la façon dont cet objectif est réalisé (par exemple, par nom, adresse, ou contexte). Ces termes ne devraient pas être pris pour l'affirmation qu'un identifiant définit ou incorpore l'identité de ce qui est référencé, bien que cela puisse être le cas de certains identifiants. Pas plus qu'on ne devrait supposer qu'un système utilisant des URI va accéder aux ressources identifiées : dans de nombreux cas, les URI sont utilisés pour noter des ressources sans aucune intention d'y accéder. De même, la ressource "un" identifiée peut n'être pas unique

par nature (par exemple, une ressource peut être un ensemble ou une transposition désigné qui varie dans le temps).

Un URI est un identifiant qui consiste en une séquence de caractères qui satisfont à la règle syntaxique nommée <URI> à la Section 3. Il permet une identification uniforme de ressources via un ensemble extensible de schémas de nommage définis séparément (paragraphe 3.1). La façon dont cette identification est accomplie, allouée, ou permise est renvoyée à chaque spécification de schéma.

La présente spécification ne met aucune limite à la nature d'une ressource, sur les raisons pour lesquelles une application peut chercher à se référer à une ressource, ou sur les sortes de systèmes qui peuvent utiliser des URI dans le but d'identifier des ressources. La présente spécification n'exige pas qu'un URI persiste à identifier la même ressource à tout jamais, bien que ce soit un but commun à tous les schémas d'URI. Quoi qu'il en soit, rien dans la présente spécification n'empêche une application de se limiter elle-même à un type particulier de ressources, ou à un sous-ensemble d'URI qui conservent les caractéristiques désirées par cette application.

Les URI ont une portée mondiale et sont interprétés de façon cohérente, sans considération du contexte, bien que le résultat de cette interprétation puisse être en rapport avec le contexte de l'utilisateur final. Par exemple, "http://localhost/" a la même interprétation pour tout utilisateur de cette référence, même si l'interface de réseau qui correspond à "localhost" peut être différente pour chaque utilisateur final : l'interprétation est indépendante de l'accès. Cependant, une action faite sur la base de cette référence aura lieu en rapport avec le contexte de l'utilisateur final, ce qui implique qu'une action destinée à se référer à une chose unique au monde doit utiliser un URI qui distingue cette ressource de toutes les autres choses. Les URI qui identifient une relation avec le contexte local de l'utilisateur final ne devraient être utilisés que lorsque le contexte lui-même est un aspect de la définition de la ressource, comme lorsqu'un manuel d'aide en ligne se réfère à un fichier du système de fichiers de l'utilisateur (par exemple, "file:///etc/hosts").

1.1.1. Syntaxe générique

Chaque URI commence par un nom de schéma, comme défini au paragraphe 3.1, qui se réfère à une spécification pour l'allocation des identifiants au sein de ce schéma. Comme telle, la syntaxe d'URI est un système de nommage fédéré et extensible au sein duquel chaque spécification de schéma peut encore restreindre la syntaxe et la sémantique des identifiants qui utilisent ce schéma.

La présente spécification définit les éléments de la syntaxe d'URI qui sont nécessaires pour tous les schémas d'URI ou sont communs à de nombreux schémas d'URI. Elle définit donc la syntaxe et la sémantique nécessaire pour mettre en œuvre un mécanisme d'analyse des références d'URI indépendant du schéma, par lequel le traitement d'un URI, dépendant du schéma, peut être différé jusqu'à ce que la sémantique dépendante du schéma devienne nécessaire. De même, les protocoles et les formats de données qui utilisent les références d'URI peuvent se reporter à la présente spécification pour la définition de la portée de la syntaxe autorisée pour tous les URI, y compris les schémas qui doivent encore être définis. Ceci découple l'évolution des schémas d'identification de l'évolution des protocoles, des formats de données, et des mises en œuvre qui utilisent les URI.

Un analyseur de syntaxe générique d'URI peut analyser toute référence d'URI dans ses composants majeurs. Une fois que le schéma est déterminé, il peut effectuer une analyse spécifique du schéma sur les composants. En d'autres termes, la syntaxe générique d'URI est un surensemble de la syntaxe de tous les schémas d'URI.

1.1.2. Exemples

Les exemples d'URI suivants illustrent plusieurs schémas d'URI avec des variantes dans les composants de syntaxe communs :

```
ftp://ftp.is.co.za/rfc/rfc1808.txt
http://www.ietf.org/rfc/rfc2396.txt
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/
urn:oasis:names:specification:docbook:dtd:xml:4.1.2
```

1.1.3. URI, URL, et URN

Un URI peut encore être classé comme localiseur, comme nom, ou les deux. Le terme "Localiseur uniforme de ressource (*Uniform Resource Locator*)" (URL) se réfère à l'ensemble des URI qui, en plus de l'identification de ressource, donnent le moyen de localiser la ressource en décrivant son mécanisme d'accès primaire (par exemple, sa "localisation " réseau). Le terme de "Nom uniforme de ressource (*Uniform Resource Name*)" (URN) a été utilisé dans le passé pour se référer aux deux URI sous le schéma "urn" [RFC2141], qui doivent rester uniques au monde et persister même lorsque la ressource cesse d'exister ou devient indisponible, ainsi qu'à tout autre URI ayant les propriétés d'un nom.

Un schéma individuel n'a pas à être classé comme "nom" ou "localiseur". Des URI provenant de tout schéma donné peuvent avoir les caractéristiques de noms ou de localiseurs ou des deux, ce qui dépend souvent de la persévérance et de l'attention des autorités de nommage dans l'allocation des identifiants, plutôt que de la qualité du schéma. A l'avenir, les spécifications et la documentation qui s'y rapporte devraient utiliser le terme général d'"URI" plutôt que les termes plus restrictifs d'"URL" et d'"URN" [RFC3305].

1.2. Considérations sur la conception

1.2.1. Transcription

La syntaxe d'URI a été conçue en considération principale de la transcription mondiale. Un URI est une séquence de caractères provenant d'un ensemble très limité : les lettres de l'alphabet latin de base, les chiffres, et quelques caractères spéciaux. Un URI peut être représenté de différentes façons ; par exemple, de l'encre sur du papier, des pixels sur un écran ou une séquence d'octets de codage de caractères. L'interprétation d'un URI ne dépend que des caractères utilisés et pas de la façon dont ces caractères sont représentés dans un protocole réseau.

L'objectif de la transcription peut être décrit par un scénario simple. Imaginez deux collègues, Sam et Kim, assis dans un bar à une conférence internationale et échangeant des idées de recherches. Sam demande à Kim une localisation où obtenir plus d'informations, et Kim écrit l'URI du site de recherche sur une serviette en papier. A son retour chez lui, Sam sort la serviette et tape l' URI sur son ordinateur, qui restitue alors les informations auxquelles Kim faisait référence.

Ce scénario met en lumière plusieurs considérations sur la conception.

O Un URI est une séquence de caractères qui n'est pas toujours représentée comme séquence d'octets.

O Un URI peut être transcrit à partir d'une source non-réseau et donc devrait consister en caractères qui sont le plus vraisemblablement capables d'être entrés dans un ordinateur, compte tenu des contraintes imposées par les claviers (et les dispositifs d'entrée qui s'y rapportent) dans les divers langages et dialectes.

O Un URI doit souvent être mémorisé par les gens, et il est plus facile de mémoriser un URI quand il consiste en composants significatifs ou familiers.

Ces considérations sur la conception ne sont pas toujours compatibles. Par exemple, c'est souvent que le nom de plus parlant pour un composant d'URI exige des caractères qui ne peuvent être tapés dans certains systèmes. La capacité à transcrire un identifiant de ressource d'un support à un autre a été considérée comme plus importante que d'avoir un URI ayant les composants les plus parlants.

Dans des contextes locaux ou régionaux, et avec l'amélioration des technologies, les utilisateurs pourraient tirer profit de la possibilité d'utiliser une plus large gamme de caractères ; un tel usage n'est pas défini dans la présente spécification. Les octets codés en pourcentage (paragraphe 2.1) peuvent être utilisés au sein d'un URI pour représenter des caractères en dehors de la gamme de l'ensemble des caractères codés en US-ASCII si cette représentation est permise par le schéma ou l'élément de protocole dans lequel est référencé l'URI. Une telle définition devrait spécifier le codage de caractère utilisé pour transposer ces caractères en octets avant d'être codés en pourcentage pour l'URI.

1.2.2. Séparation de l'identification de l'interaction

Une erreur courante au sujet des URI est qu'ils ne seraient utilisés que pour se référer à des ressources accessibles. L'URI par lui-même ne donne que l'identification ; l'accès à la ressource n'est ni garanti ni

impliqué par la présence d'un URI. En fait, toute opération associée à une référence d'URI est définie par l'élément de protocole, l'attribut de format des données, ou un texte en langage naturel dans lequel il apparaît.

Pour un URI donné, un système peut essayer d'effectuer diverses opérations sur la ressource, qui peuvent être caractérisées par des mots tels que "accéder", "mettre à jour", "remplacer", ou "trouver les attributs". De telles opérations sont définies par les protocoles qui utilisent les URI, et non par la présente spécification. Cependant, on utilise effectivement quelques termes généraux pour décrire les opérations courantes sur les URI. La "résolution" d'URI est le processus de détermination d'un mécanisme d'accès et des paramètres appropriés nécessaires pour dé référencer un URI ; cette résolution peut nécessiter plusieurs itérations. Pour utiliser ce mécanisme d'accès et effectuer une action sur la ressource de l'URI il faut "dé référencer" l'URI.

Lorsque des URI sont utilisés au sein de système de restitution d'information pour identifier les sources de l'information, la forme la plus courante de dé référencement d'URI est la "restitution" : l'utilisation d'un URI pour restituer une représentation de sa ressource associée. "représentation" est une séquence d'octets, conjointe avec des métadonnées de représentation qui décrivent ces octets, qui constitue un enregistrement de l'état de la ressource à l'instant de la génération de la représentation. La restitution est achevée par un processus qui peut inclure l'utilisation de l'URI comme clé cachée pour vérifier une représentation cachée localement, la résolution de l'URI pour déterminer un mécanisme d'accès approprié (s'il en est), et le dé référencement de l'URI pour les besoins de l'application de l'opération de restitution. Selon le protocole utilisé pour effectuer la restitution, des informations supplémentaires peuvent être fournies au sujet de la ressource (métadonnées de ressource) et de ses relations avec d'autres ressources.

Les références d'URI dans les systèmes de restitution d'informations sont conçues pour être à corrélation différée : le résultat d'un accès est généralement déterminé lorsqu'il est effectué et il peut varier dans le temps ou à cause d'autres aspects de l'interaction. Ces références sont créées pour être utilisées à l'avenir : ce qui est identifié n'est pas un résultat spécifique obtenu dans le passé, mais plutôt des caractéristiques dont on espère qu'elles seront vraies pour des résultats futurs. Dans de tels cas, la ressource à laquelle l'URI fait référence est réellement une uniformité de caractéristiques telles qu'observées au cours du temps, qui peuvent être élucidées par des commentaires ou assertions supplémentaires faits par le fournisseur de la ressource.

Bien que de nombreux schémas d'URI soient nommés d'après des protocoles, cela n'implique pas que l'utilisation de ces URI donnera accès à la ressource via le protocole désigné. Les URI sont souvent simplement utilisés pour les besoins de l'identification. Même lorsqu'un URI est utilisé pour restituer une représentation d'une ressource, cet accès peut se faire à travers des passerelles, des mandataires, des mémoires cachées, et des services de résolution de nom qui sont indépendants du protocole associé au nom du schéma. La résolution de certains URI peut exiger l'utilisation de plus d'un protocole (par exemple, DNS et HTTP sont normalement tous deux utilisés pour accéder à un serveur d'origine "http" d'un URI lorsqu'une représentation n'est pas trouvée dans une mémoire cachée locale).

1.2.3. Identifiants hiérarchiques

La syntaxe d'URI est organisée de façon hiérarchique, avec des composants listés en ordre de signification décroissante de gauche à droite. Pour certains schémas d'URI, la hiérarchie visible est limitée au schéma lui-même : tout ce qui est après le délimiteur de composant de schéma (":") est considéré comme opaque pour le traitement de l'URI. D'autres schémas d'URI rendent la hiérarchie explicite et visible pour les algorithmes d'analyse génériques.

La syntaxe générique utilise les caractères barre oblique ("/"), point d'interrogation ("?"), et le signe dièse ("#") pour délimiter les composants qui sont significatifs pour l'interprétation hiérarchique de l'analyseur générique d'un identifiant. En plus d'une aide à la lisibilité de tels identifiants grâce à l'utilisation cohérente d'une syntaxe familière, cette représentation uniforme de la hiérarchie à travers les schémas de nommage permet de faire des références indépendantes du schéma par rapport à cette hiérarchie.

Il arrive souvent qu'un groupe ou "arborescence" de documents ait été construite pour servir un objectif commun, et que la grande majorité des références d'URI dans ces documents visent des ressources au sein de l'arborescence plutôt qu'à l'extérieur. De même, des documents situés sur un site particulier vont plus vraisemblablement se référer à d'autres ressources de ce site plutôt qu'à des ressources de sites distants. Le référencement relatif des URI permet aux arborescences de document d'être partiellement indépendantes de leur localisation et de leur schéma d'accès. Par exemple, il est possible qu'un unique ensemble de documents hypertexte soit simultanément accessible et traversable via chacun des schémas "file", "http", et

"ftp" si les documents se réfèrent l'un à l'autre avec des références croisées. De plus, de telles arborescences de document peuvent être déplacées, comme un tout, sans changer aucune des références croisées.

Une référence croisée (paragraphe 4.2) se réfère à une ressource en décrivant les différences au sein d'un espace de nom hiérarchique entre le contexte de la référence et l'URI cible. L'algorithme de résolution de référence, présenté à la Section 5, définit comment une telle référence est transformée en URI cible. Comme les références croisées ne peuvent être utilisées que dans le contexte d'URI hiérarchiques, les concepteurs de nouveaux schémas d'URI devraient utiliser une syntaxe cohérente avec les composants hiérarchiques de la syntaxe générique à moins qu'il n'y aient des raisons majeurs qui interdisent des références croisées au sein de ce schéma.

NOTE : Les spécifications précédentes utilisaient les termes " URI partiel" et "URI relative" pour noter une référence croisée à un URI. Comme certains lecteurs méconnaissaient ces termes en comprenant que les URI relatives étaient un sous-ensemble d'URI plutôt qu'une méthode de référencement des URI, la présente spécification se réfère simplement à eux comme références croisées.

Toutes les références d'URI sont analysées par des analyseurs de syntaxe génériques lorsqu'ils sont utilisés. Cependant, puisque le traitement hiérarchique n'a pas d'effet sur un URI absolu dans une référence à moins qu'il ne contienne un ou plusieurs segments point (des segments de chemin complet de "." ou "..", comme décrit au paragraphe 3.3), les spécifications de schéma d'URI peuvent définir des identifiants opaques en interdisant l'utilisation des caractères barre oblique, point d'interrogation, et les URI "scheme:." et "scheme:..".

1.3. Notation de la syntaxe

La présente spécification utilise la notation Augmented Backus-Naur Form (ABNF) de la [RFC2234], y compris les règles de syntaxe ABNF centrales suivantes définies par cette spécification : ALPHA (lettres), CR (retour chariot), DIGIT (chiffres décimaux), DQUOTE (double guillemets), HEXDIG (chiffres hexadécimaux), LF (retour à la ligne), et SP (espace). La syntaxe URI complète est rassemblée dans l'Appendice A.

2. Caractères

La syntaxe d'URI donne une méthode de codage des données, a priori dans le but d'identification d'une ressource, comme une séquence de caractères. Les caractères d'un URI sont, à leur tour, fréquemment codés comme des octets pour le transport ou la présentation. La présente spécification ne dédie aucun codage de caractère particulier pour la transposition entre les caractères de l'URI et les octets utilisés pour stocker ou transmettre ces caractères. Lorsqu'un URI apparaît dans un élément de protocole, le codage du caractère est défini par ce protocole; sans une telle définition, un URI est supposé être dans le même codage de caractères que le texte environnant.

La notation ABNF définit ses valeurs terminales comme entiers non négatifs (points de code) fondés sur l'ensemble de caractères codés de l'ASCII US [ASCII]. Comme un URI est une séquence de caractères, on doit inverser cette relation afin de comprendre la syntaxe de l'URI. Les valeurs d'entier utilisées par l'ABNF doivent donc être retransposées dans leurs caractères correspondants via l'US-ASCII afin de mener à bien les règles syntaxiques.

Un URI se compose d'un ensemble limité de caractères consistant en chiffres, lettres, et de quelques symboles graphiques. Un sous-ensemble réservé de ces caractères peut être utilisé pour délimiter les composants syntaxiques au sein d'un URI alors que les caractères restants, y compris à la fois l'ensemble non réservé et les caractères réservés qui ne jouent pas le rôle de délimiteurs, définissent les données d'identification de chaque composant.

2.1. Codage en pourcentage

Un mécanisme de codage en pourcentage est utilisé pour représenter un octet de données dans un composant lorsque le caractère correspondant à cet octet se trouve hors de l'ensemble autorisé ou est utilisé comme délimiteur du composant au sein du composant. Un octet codé en pourcentage est codé comme un triplet de caractères, comportant le caractère pourcentage "%" suivi des deux chiffres hexadécimaux qui représentent la valeur numérique de cet octet. Par exemple, "%20" est le codage en pourcentage de l'octet binaire "00100000" (ABNF: %x20), qui en US-ASCII correspond au caractère espace (SP). Le paragraphe 2.4 décrit quand sont appliqués le codage et décodage en pourcentage.

pct-encoded = "%" HEXDIG HEXDIG

Les chiffres hexadécimaux majuscules 'A' à 'F' sont respectivement équivalents aux chiffres minuscules 'a' à 'f'. Si deux URI diffèrent seulement sur la casse des chiffres hexadécimaux utilisés dans les octets codés en pourcentage, ils sont équivalents. Pour la cohérence, les producteurs et normalisateurs d'URI devraient utiliser des chiffres hexadécimaux majuscules pour tous les codages en pourcentage.

2.2. Caractères réservés

Les URI comportent des composants et sous-composants qui sont délimités par des caractères dans l'ensemble "réservé". Ces caractères sont appelés "réservés" parce qu'ils peuvent être (ou n'être pas) définis comme délimiteurs par la syntaxe générique, par chaque syntaxe spécifique de schéma, ou par la syntaxe spécifique de la mise en œuvre d'un algorithme de dé référencement d'URI. Si les données pour un composant d'URI doivent entrer en conflit avec la destination de délimiteur d'un caractère réservé, les données en conflit doivent être codées en pourcentage avant la formation de l'URI.

réservé = gen-delims / sub-delims

gen-delims = ":" / "/" / "?" / "#" / "[" / "]" / "@"

sub-delims = "!" / "$" / "&" / "'" / "(" / ")"
/ "*" / "+" / "," / ";" / "="

L'objet des caractères réservés est de fournir un ensemble de caractères de délimitation qu'on puisse distinguer des autres données au sein d'un URI. Les URI qui diffèrent dans le remplacement d'un caractère réservé par l'octet codé en pourcentage correspondant ne sont pas équivalents. Le codage en pourcentage d'un caractère réservé, ou le décodage d'un octet codé en pourcentage qui correspond à un caractère réservé va changer la façon dont l'URI est interprété par la plupart des applications. Et donc, les caractères de l'ensemble réservé sont protégés contre la normalisation et peuvent donc être utilisés en toute sécurité par les algorithmes spécifiques d'un schéma ou spécifiques d'un producteur pour les sous-composants de données de délimitation au sein d'un URI.

Un sous ensemble de caractères réservés (gen-delims) est utilisé comme délimiteurs des composants génériques d'URI décrits à la Section 3. Une règle syntaxique ABNF de composant n'utilisera pas les noms de règle réservée ou gen-delims directement ; au lieu de cela, chaque règle de syntaxe fait la liste des caractères autorisés au sein de ce composant (c'est-à-dire, qui ne le délimitent pas), et tout caractère parmi ceux-ci qui est aussi dans l'ensemble réservé est "réservé" pour être utilisé comme délimiteur de sous-composant au sein du composant. Seuls les sous-composants les plus courants sont définis dans la présente spécification ; les autres sous-composants peuvent être définis par une spécification de schéma d'URI, ou par la syntaxe spécifique de la mise en œuvre d'un algorithme de dé référencement d'URI, pourvu que de tels sous-composants soient délimités par des caractères de l'ensemble réservé admis au sein de ce composant.

Un URI produisant des applications devrait coder en pourcentage les octets de données qui correspondent aux caractères de l'ensemble réservé à moins que ces caractères ne soient spécifiquement admis par le schéma d'URI pour représenter des données dans ce composant. Si on trouve un caractère réservé dans un composant d' URI et qu'on ne trouve pas de rôle de délimiteur pour ce caractère, il doit alors être interprété comme représentant l'octet de données correspondant au codage de ce caractère en US-ASCII.

2.3. Caractères non réservés

Les caractères qui sont permis dans un URI mais n'ont pas un objet réservé sont appelés non réservés. Ils comportent les lettres majuscules et minuscules, les chiffres décimaux, le tiret, le point, le souligné et le tilde.

Non réservé = ALPHA / DIGIT / "-" / "." / "_" / "~"

Les URI qui diffèrent par le remplacement d'un caractère non réservé de l'octet US-ASCII correspondant codé en pourcentage sont équivalents : ils identifient la même ressource. Cependant, les mises en œuvre de comparaisons d'URI n'effectuent pas toujours de normalisation avant la comparaison (voir la Section 6). Dans un souci de cohérence, les octets codés en pourcentage dans la gamme ALPHA (%41-%5A et61-%7A), DIGIT (%30-%39), tiret (%2D), point (%2E), souligné (%5F), ou tilde (%7E) ne devraient pas être créés par

les producteurs d'URI, et lorsqu'ils apparaissent dans un URI, devraient être décodés comme caractères réservés correspondants par les normalisateurs d'URI.

2.4. Quand coder ou décoder

Dans des circonstances normales, le seul moment où des octets sont codés en pourcentage dans un URI est durant le processus de production de l'URI à partir de ses composants. Cela se produit lorsque une mise en œuvre détermine quels sont les caractères réservés qui doivent être utilisés comme délimiteurs de sous-composants et ceux qui peuvent être utilisés en toute sécurité comme données. Une fois produit, un URI est toujours sous sa forme codée en pourcentage.

Lorsqu'un URI est dé référencé, les composants et sous-composants significatifs pour le processus de dé référencement spécifique du schéma (s'il en est) doivent être analysés et séparés avant que les octets codés en pourcentage au sein de ces composants puissent être décodés en toute sécurité, car autrement, les données pourraient être confondues avec des délimiteurs de composant. La seule exception est celle des octets codés en pourcentage correspondant aux caractères de l'ensemble non réservé, qui peuvent être décodés à tout moment. Par exemple, l'octet correspondant au caractère tilde ("~") est souvent codé comme "%7E" par les anciennes mises en œuvre de traitement d'URI ; le "%7E" peut être remplacé par "~" sans changer son interprétation.

Comme le caractère pourcentage ("%") sert d'indicateur des octets codés en pourcentage, il doit être codé en pourcentage avec "%25" pour cet octet afin d'être utilisé comme donnée au sein d'un URI. Les mises en œuvre ne doivent pas coder ou décoder en pourcentage plus d'une fois la même chaîne, car décoder une chaîne déjà décodée peut conduire à mal interpréter un octet de données de pourcentage comme le début d'un codage en pourcentage, ou vice versa dans le cas du codage en pourcentage d'une chaîne déjà codée en pourcentage.

2.5. Données d'identification

Les caractères de l'URI donnent les données d'identification de chacun des composants de l'URI, et servent d'interface externe pour l'identification entre systèmes. Bien que la présence et la nature de l'interface de production d'URI soient cachées aux clients qui utilisent ces URI (et est en dehors du domaine d'application des exigences d'interopérabilité définies par la présente spécification), c'est une fréquente source de confusion et d'erreurs d'interprétation des questions de caractère des URI. Ceux qui réalisent les mises en œuvre doivent savoir qu'il y a de nombreux codages de caractères qui sont impliqués dans la production et la transmission des URI : nom local et codage de données, codage d'interface publique, codage de caractère d'URI, codage de format de données, et codage de protocole.

Les noms locaux, comme les noms de système de fichier, sont mémorisés avec un codage de caractère local. Les applications qui produisent des URI (par exemple, les serveurs d'origine) vont normalement utiliser le codage local comme base de production de noms significatifs. Le producteur d'URI va transformer le codage local en un codage convenable pour une interface publique puis transformer le codage d'interface publique avec l'ensemble restreint de caractères d'URI (réservé, non réservé, et codages en pourcentage). Ces caractères sont à leur tour codés comme octets à utiliser comme référence dans un format de données (par exemple, un charset de document), et de tels formats de données sont souvent codés ultérieurement pour transmission selon des protocoles Internet.

Dans la plupart des systèmes, l'apparition d'un caractère réservé dans un composant d'URI est interprétée comme la représentation de l'octet de données correspondant au codage de ce caractère en US-ASCII. Les utilisateurs d'URI supposent que la lettre "X" correspond à l'octet "01011000", et même lorsque cette supposition est incorrecte, il n'y a pas de mal à la faire. Un système qui fournit des identifiants en interne sous la forme d'un codage de caractères différent, comme EBCDIC, va généralement effectuer une traduction de caractères des identifiants textuels en UTF-8 [STD63] (ou autre surensemble de codage de caractères US-ASCII) à une interface interne, produisant par là plus d'identifiants significatifs que ce qui résulte du simple codage en pourcentage des octets d'origine.

Par exemple, considérons un service d'information qui fournit des données, mémorisées localement en utilisant un système de fichiers fondé sur EBCDIC, à des clients sur l'Internet au moyen d'un serveur HTTP. Lorsqu'un auteur crée un fichier du nom de "Laguna Beach" dans ce système de fichiers, l'URI "http" correspondant à cette ressource est censé contenir la chaîne signifiant "Laguna%20Beach". Si cependant, ce serveur produit des URI en utilisant une transposition d'octets bruts trop simpliste, cela pourrait donner un URI contenant "%D3%81%87%A4%95%81@%C2%85%81%83%88". Une interface de transcodage interne

résout ce problème en transcodant le nom local en un super-ensemble US-ASCII avant de produire l'URI. Naturellement, l'interprétation appropriée d'un URI entrant sur une telle interface exige le décodage des octets codés en pourcentage (par exemple, "%20" en SP) avant que le transcodage inverse ne soit appliqué pour obtenir le nom local.

Dans certains cas, l'interface interne entre un composant d'URI et les données d'identification qu'il doit représenter est beaucoup moins directe qu'une traduction de codage de caractères. Par exemple, des portions d'un URI peuvent refléter une question sur des données non-ASCII, ou des coordonnées numériques sur une carte. De même, un schéma d'URI peut définir des composants avec des exigences de codage supplémentaires appliquées avant de former le composant et produire l'URI.

Lorsqu'un nouveau schéma d'URI définit un composant qui représente des données textuelles consistant en caractères provenant de l'ensemble universel de caractères [UCS, *Universal Character Set*], les données devraient d'abord être codées comme octets conformément au codage de caractères UTF-8 [STD63] ; alors seulement, les octets qui ne correspondent pas aux caractères de l'ensemble non réservé devraient être codés en pourcentage. Par exemple, le caractère A serait représenté par "A", le caractère LETTRE LATINE MAJUSCULE ACCENT GRAVE serait représentée par "%C3%80", et le caractère LETTRE KATAKANA A serait représenté par "%E3%82%A2".

3. Composants syntaxiques

La syntaxe générique d' URI consiste en séquences hiérarchiques de composants appelés scheme (*schéma*), authority (*autorité*), path *(chemin*), query (*question*), et fragment.

```
URI         = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part   = "//" authority path-abempty
            / path-absolute
            / path-rootless
            / path-empty
```

Les composants schéma et chemin sont nécessaires, bien que le chemin puisse être vide (pas de caractères). Lorsque *autorité* est présent, le chemin doit être vide ou commencer par un caractère barre oblique ("/"). Lorsque *autorité* n'est pas présent, le chemin ne peut pas commencer par deux caractères barre oblique ("//"). Ces restrictions donnent cinq règles ABNF différentes pour un chemin (paragraphe 3.3), dont une seule correspondra à toute référence d'URI donnée.

Ci-après figurent deux exemples d'URI et de leurs composants :

```
  foo://example.com:8042/over/there?name=ferret#nose
  \_/   \______________/\_________/ \_________/ \__/
   |           |            |            |        |
scheme     authority       path        query   fragment
   |   _____________________|__
  / \ /                     \
  urn:example:animal:ferret:nose
```

3.1. Schéma

Chaque URI commence par un nom de schéma qui se réfère à une spécification pour l'allocation des identifiants au sein de ce schéma. Comme telle, la syntaxe d'URI est un système fédéré et extensible de nommage au sein duquel chaque spécification de schéma peut encore restreindre la syntaxe et la sémantique des identifiants utilisant ce schéma.

Les noms de schéma consistent en une séquence de caractères commençant par une lettre suivie par toute combinaison de lettres, chiffres, plus ("+"), point ("."), ou tiret ("-"). Bien que les schémas soient insensibles à la casse, la forme canonique est en minuscules et les mises en œuvre devraient accepter les lettres majuscules comme équivalentes aux minuscules dans les noms de schéma (par exemple, permettre"HTTP" aussi bien que "http") pour des besoins d'efficacité mais ne devraient produire que des schémas de noms en minuscules pour la cohérence.

scheme = ALPHA *(ALPHA / DIGIT / "+" / "-" / ".")

Les schémas individuels ne sont pas spécifiés dans le présent document. Le processus pour l'enregistrement des nouveaux schémas d'URI est défini à part par le [BCP35]. Le registre des schémas assure la maintenance de la transposition entre les noms de schéma et leurs spécifications. On trouvera des conseils pour les concepteurs de nouveaux schémas d'URI dans la [RFC2718]. Les spécifications de schéma d'URI doivent définir leur propre syntaxe de sorte que toute chaîne correspondant à sa syntaxe spécifique de schéma corresponde aussi à la grammaire <absolute-URI>, telle que décrite au paragraphe 4.3.

En présence d'un URI qui viole une ou plusieurs contraintes spécifiques de schéma, le processus de résolution spécifique de schéma devrait afficher la référence comme erronée plutôt que d'ignorer les parties non utilisées ; en faisant ainsi, on réduit le nombre d'URI équivalents et on aide à détecter les violations de syntaxe générique, qui peuvent indiquer si l'URI a été construit pour induire l'usager en erreur (paragraphe 7.6).

3.2. Autorité

De nombreux schémas d'URI incluent un élément hiérarchique d'autorité de nommage, de sorte que la gouvernance de l'espace de nom définie par le reste de l'URI est déléguée à cette autorité (qui peut, à son tour, subdéléguer). La syntaxe générique donne un moyen usuel de distinguer une autorité fondée sur un nom enregistré ou sur une adresse de serveur, ainsi que les ports facultatifs et les informations d'utilisateur.

Le composant autorité est précédé d'une double barre oblique ("//") et se termine sur le caractère barre oblique ("/"), point d'interrogation ("?"), ou dièse ("#") suivant, ou par la fin de l'URI.

authority = [userinfo "@"] host [":" port]

Les producteurs et normalisateurs d'URI devraient omettre le délimiteur ":" qui sépare l'hôte du port si le composant port est vide. Certains schémas n'admettent pas de sous-composant userinfo et/ou port.

Si un URI contient un composant autorité, le composant chemin (*path*) doit être vide ou commencer par un caractère barre oblique ("/"). Les analyseurs qui ne font pas de validation (ceux qui séparent simplement une référence d'URI selon ses principaux composants) ignoreront souvent la structure de sous-composant de l'autorité, la traitant comme une chaîne opaque depuis la double barre oblique jusqu'au premier délimiteur de terminaison, jusqu'au moment où l'URI sera dé référencé.

3.2.1. Informations d'utilisateur

Le sous-composant userinfo (*informations d'utilisateur*) peut comporter un nom d'utilisateur et, facultativement, des informations spécifiques du schéma sur la façon d'obtenir l'autorisation d'accès à la ressource. Les informations d'utilisateur, si elles sont présentes, sont suivies par l'arobase ("@") qui les sépare de l'hôte.

userinfo = *(unreserved / pct-encoded / sub-delims / ":")

L'utilisation du format "user:password" dans le champ userinfo est déconseillée. Les applications ne devraient pas restituer en texte clair toutes données figurant après le premier caractère deux points (":") trouvé après un sous-composant userinfo à moins que les données figurant après les deux points ne soient la chaîne vide (qui indique qu'il n'y a pas de mot de passe). Les applications peuvent choisir d'ignorer ou rejeter de telles données lorsqu'elles sont reçues au titre d'une référence et devraient rejeter la mémorisation de telles données sous une forme non cryptée. La transmission d'informations d'authentification en texte clair s'est révélée un risque pour la sécurité dans presque tous les cas où elle a été utilisée.

Les applications qui traduisent un URI pour les besoins de l'information en retour de l'utilisateur, comme dans une navigation en hypertexte graphique, devraient traduire les userinfo d'une façon qui les distingue du reste d'un URI, lorsque c'est faisable. Une telle traduction aidera l'utilisateur dans les cas où les userinfo ont été bâties de façon à ressembler à un nom de domaine de confiance (paragraphe 7.6).

3.2.2. Hôte

Le sous-composant host (*hôte*) de authority est identifié par un littéral IP compris entre crochets, une adresse IPv4 en forme décimale avec point, ou un nom enregistré. Le sous-composant host est insensible à la casse. La présence d'un sous-composant host au sein d'un URI n'implique pas que le schéma requière que l'accès à cet hôte se fasse par l'Internet. Dans de nombreux cas, la syntaxe host n'est utilisée que pour les besoins de la réutilisation du processus d'enregistrement existant créé et développé pour DNS, et obtenir donc ainsi un nom unique au monde sans avoir le coût de développement d'un autre registre. Cependant, une telle utilisation porte son propre coût : la propriété d'un nom de domaine peut changer de mains avec le temps pour des raisons non anticipées par le producteur de l'URI. Dans d'autres cas, les données au sein du composant host identifient un nom enregistré qui n'a rien à voir avec un hôte Internet. On utilise le nom "host" pour la règle ABNF parce que c'est l'objet le plus commun, mais ce n'est pas son seul objet.

```
host       = IP-literal / IPv4address / reg-name
```

La règle de syntaxe pour host est ambiguë parce qu'elle ne fait pas complètement la distinction entre une adresse IPv4 et un reg-name. Pour rendre la syntaxe non ambiguë, on applique l'algorithme "le premier qui correspond l'emporte" : si host satisfait à la règle pour l'adresse IPv4, il devrait alors être considéré comme une adresse IPv4 littérale et non un reg-name. Bien que host soit insensible à la casse, les producteurs et normalisateurs devraient utiliser les minuscules pour les noms enregistrés et les adresses en hexadécimal pour les besoins de l'harmonisation, et n'utiliser les lettres majuscules que pour les codages en pourcentage.

Un hôte identifié par une adresse littérale de Protocole Internet, version 6 [RFC3513] ou plus récente, se distingue en enfermant le littéral IP entre crochets ("[" et "]"). C'est le seul endroit dans la syntaxe d'URI où des caractères crochets sont admis. En prévision du futur, et des formats d'adresse littérale IP non encore définis, une mise en œuvre peut utiliser un fanion de version facultatif pour indiquer explicitement un tel format plutôt que de compter sur une détermination heuristique.

```
IP-literal = "[" ( IPv6address / IPvFuture  ) "]"

IPvFuture  = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )
```

Le fanion de version n'indique pas la version IP ; il indique plutôt les versions futures du format littéral. Comme telles, les mises en œuvre ne doivent pas fournir le fanion de version pour les formes d'adresse littérale IPv4 et IPv6 décrites ci-dessous. Si un URI contenant une adresse IP littérale qui commence par "v" (quelque soit la casse), indiquant que le fanion de version est présent, est dé référencé par une application qui ne connaît pas la signification de ce fanion de version, l'application devrait alors retourner une erreur appropriée pour "mécanisme d'adresse non accepté".

Un hôte identifié par une adresse littérale IPv6 est représenté entre crochets sans fanion de version devant. L'ABNF fourni ici est une traduction de la définition textuelle d'une adresse littérale IPv6 fournie dans la [RFC3513]. Cette syntaxe ne prend pas en charge les identifiants de zone d'adressage du domaine IPv6.

Une adresse IPv6 de 128 bits se divise en huit morceaux de 16 bits. Chaque morceau est représenté numériquement en hexadécimal insensible à la casse, en utilisant un à quatre chiffres hexadécimaux (les zéros en tête sont permis). Les huit morceaux codés sont donnés avec celui de plus fort poids en premier, séparés par des caractères deux points. Facultativement, les deux morceaux de plus faible poids peuvent être représentés en format textuel d'adresse IPv4. Une séquence d'un ou plusieurs morceaux de 16 bits consécutifs de valeur zéro peut être élidées au sein de l'adresse, en omettant tous les chiffres et en laissant exactement deux caractères deux points consécutifs à leur place pour marquer l'élision.

```
IPv6address =                            6( h16 ":" ) ls32
            /                       "::" 5( h16 ":" ) ls32
            / [               h16 ] "::" 4( h16 ":" ) ls32
            / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
            / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
            / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
            / [ *4( h16 ":" ) h16 ] "::"              ls32
            / [ *5( h16 ":" ) h16 ] "::"              h16
            / [ *6( h16 ":" ) h16 ] "::"

ls32        = ( h16 ":" h16 ) / IPv4address
            ; 32 bits de plus faible poids de l'adresse
```

```
h16       = 1*4HEXDIG
          ; 16 bits de l'adresse représentés en hexadécimal
```

Un hôte identifié par une adresse littérale IPv4 est représenté en notation décimale à point (une séquence de quatre nombres décimaux dans la gamme de 0 à 255, séparés par "."), comme décrit dans la [RFC1123] par référence à [RFC0952]. Noter que d'autres formes de notation à point peuvent être interprétées sur certaines plates-formes, comme décrit au paragraphe 7.4, mais seule la forme décimale à point de quatre octets est permise par la présente grammaire.

```
IPv4address = dec-octet "." dec-octet "." dec-octet "." dec-octet

dec-octet   = DIGIT               ; 0-9
            / %x31-39 DIGIT       ; 10-99
            / "1" 2DIGIT          ; 100-199
            / "2" %x30-34 DIGIT   ; 200-249
            / "25" %x30-35        ; 250-255
```

Un hôte identifié par un nom enregistré est une séquence de caractères habituellement destinée à faire des recherches au sein d'un hôte défini localement ou d'un registre de nom de service, bien que la sémantique spécifique de schéma d'URI puisse exiger qu'un registre spécifique (ou un tableau de noms fixés) soit utilisé à la place. Le mécanisme de registre de nom le plus courant est l'annuaire des domaines (DNS, *Domain Name System*). Un nom enregistré destiné aux recherches dans le DNS utilise la syntaxe définie au paragraphe 3.5 de la [RFC1034] et au paragraphe 2.1 de la [RFC1123]. Un tel nom consiste en une séquence d'étiquettes de domaines séparées par ".", chaque étiquette de domaine commençant et se terminant par un caractère alphanumérique et pouvant aussi contenir des caractères "-". L'étiquette de domaine la plus à droite d'un nom de domaine pleinement qualifié dans DNS peut être suivie par un seul "." et le devrait s'il est nécessaire de distinguer entre le nom de domaine complet et un domaine local.

```
reg-name    = *( unreserved / pct-encoded / sub-delims )
```

Si le schéma d'URI définit un hôte par défaut, cet hôte par défaut s'applique alors lorsque le sous-composant hôte est indéfini ou lorsque le nom enregistré est vide (longueur zéro). Par exemple, le schéma d'URI "file" est défini de telle sorte que pas d'autorité, un hôte vide, et "localhost" signifient tous la machine de l'utilisateur final, alors que le schéma "http" considère comme invalide une autorité absente ou un hôte vide.

La présente spécification ne définit aucune technologie de recherche de nom enregistré particulière et n'impose donc pas de contrainte sur la syntaxe de reg-name au-delà de ce qui est nécessaire pour l'interopérabilité. En revanche, elle délègue la question de la conformité à la syntaxe des noms enregistrés au système d'exploitation de chaque application effectuant de la résolution d'URI, et ce système d'exploitation décide de ce qu'il va permettre en matière d'identification d'hôte. Une mise en œuvre de résolution d'URI peut utiliser DNS, des tableaux d'hôtes, les pages jaunes, NetInfo, WINS, ou tout autre système pour la recherche des noms enregistrés. Cependant, un système de nommage à visée universelle, tel que les noms de domaine pleinement qualifiés de DNS, est nécessaire pour les URI destinés à avoir une portée mondiale. Les producteurs d'URI devraient utiliser des noms conformes à la syntaxe DNS, même lorsque l'utilisation de DNS n'est pas immédiatement apparente, et devraient limiter ces noms à une longueur inférieure à 255 caractères.

La syntaxe reg-name permet les octets codés en pourcentage afin de représenter les noms enregistrés en non-ASCII d'une façon uniforme indépendante de la technologie de résolution de nom sous-jacente. Les caractères non-ASCII doivent d'abord être codés conformément à UTF-8 [STD63], puis chaque octet de la séquence UTF-8 correspondante doit être codé en pourcentage pour être représenté comme caractère d'URI. Les applications qui produisent des URI ne doivent pas utiliser de codage en pourcentage dans un hôte à moins qu'il ne serve à représenter une séquence de caractères UTF-8. Lorsqu'un nom enregistré non-ASCII représente un nom de domaine internationalisé destiné à être résolu via le DNS, le nom doit être transformé en codage IDNA [RFC3490] avant la recherche de nom. Les producteurs d'URI devraient fournir ces noms enregistrés dans le codage IDNA, plutôt qu'un codage en pourcentage, s'ils souhaitent maximiser l'interopérabilité avec les résolveurs d'URI traditionnels.

3.2.3. Port

Le sous-composant port de authority est désigné par un numéro de port facultatif en décimal suivant host et délimité de lui par un seul caractère deux points (":").

```
port        = *DIGIT
```

Un schéma peut définir un port par défaut. Par exemple, le schéma "http" définit un port par défaut de "80", correspondant à son numéro de port TCP réservé. Le type de port désigné par le numéro de port (par exemple, TCP, UDP, SCTP) est défini par le schéma d'URI. Les producteurs et normalisateurs d'URI devraient omettre le composant port et son délimiteur ":" si port est vide ou si sa valeur serait la même que celle par défaut du schéma.

3.3. Path

Le composant path (*chemin*) contient des données, habituellement organisées sous forme hiérarchique, qui, avec les données du composant d'interrogation non hiérarchique (paragraphe 3.4), servent à identifier une ressource au sein du domaine d'application du schéma d'URI et de l'autorité de nommage (s'il en est). Le path est terminé par le premier point d'interrogation ("?") ou le caractère dièse ("#"), ou par la fin de l'URI.

Si un URI contient un composant authority, le composant path doit alors être vide ou commencer par un caractère barre oblique ("/"). Si un URI ne contient pas de composant authority, le path ne peut alors pas commencer par deux caractères barre oblique ("//"). De plus, une référence d'URI (paragraphe 4.1) peut être une référence de relative-path (*chemin relatif*), auquel cas le premier segment path ne peut pas contenir de caractère deux points (":"). L'ABNF exige cinq règles séparées pour rendre ces cas non ambigus, dont une seule va correspondre à la sous-chaîne path au sein d'une référence d'URI donnée. On utilise le terme générique "composant path" pour décrire la sous-chaîne d'URI que l'analyseur grammatical fait correspondre à l'une de ces règles.

```
path          = path-abempty    ; commence par "/" ou est vide
              / path-absolute   ; commence par "/" mais pas par "//"
              / path-noscheme   ; commence par un segment qui n'est pas deux points
              / path-rootless   ; commence par un segment
              / path-empty      ; caractères zéro

path-abempty  = *( segment "/")
path-absolute = "/" [ segment-nz *( segment "/") ]
path-noscheme = segment-nz-nc *( segment "/")
path-rootless = segment-nz *( segment "/")
path-empty    = 0<pchar>
segment       = *pchar
segment-nz    = 1*pchar
segment-nz-nc = 1*( non réservé / codé en pct / sous-délims / "@" )
              ; segment de longueur différente de zéro sans caractère deux points ":"

pchar         = non réservé / codé en pct / sous-délims / ":" / "@"
```

Un path consiste en une séquence de segments path séparés par un caractère barre oblique ("/"). Un path est toujours défini pour un URI, bien que le path défini puisse être vide (de longueur zéro). L'utilisation du caractère barre oblique pour indiquer la hiérarchie n'est exigée que lorsqu'un URI va être utilisé comme contexte pour des références croisées. Par exemple, l'URI <mailto:fred@example.com> a comme path "fred@example.com", alors que l'URI <foo://info.example.com?fred> a un path vide.

Les segments path "." et "..", connus aussi sous le nom de segments point, sont définis pour des références croisées au sein de la hiérarchie de nom de chemin. Ils sont destinés à être utilisés au début d'une référence de chemin relatif (paragraphe 4.2) pour indiquer la position relative au sein de l'arbre hiérarchique des noms. Ceci est similaire au rôle qu'ils jouent dans certaines structures de répertoire de fichiers de système d'exploitation pour indiquer respectivement le répertoire en cours et le répertoire parent. Cependant, à la différence d'un système de fichiers, ces segments point ne sont interprétés qu'au sein de la hiérarchie de chemin d'URI et sont retirés du processus de résolution (paragraphe 5.2).

A côté des segments point dans les chemins hiérarchiques, un segment path est considéré comme opaque par la syntaxe générique. Les applications qui produisent des URI utilisent souvent les caractères réservés permis dans un segment pour délimiter les sous-composants spécifiques du schéma ou spécifiques du traitement ou déréférencement. Par exemple, les caractères réservés point virgule (";") et égal ("=") sont souvent utilisés pour délimiter des paramètres et des valeurs de paramètre applicables à ce segment. Le caractère réservé virgule (",") est souvent utilisé à des fins semblables. Par exemple, un producteur d'URI

pourrait utiliser un segment tel que "name;v=1.1" pour indiquer une référence à la version 1.1 de "name", tandis qu'un autre pourrait utiliser un segment tel que "name,1.1" pour indiquer la même chose. Les types de paramètres peuvent être définis par une sémantique spécifique du schéma, mais dans la plupart des cas, la syntaxe d'un paramètre est spécifique de la mise en œuvre de l'algorithme de déréférencement de l'URI.

3.4. Query

Le composant query (*question*) contient des données non hiérarchiques qui, avec les données du composant path (paragraphe 3.3), servent à identifier une ressource au sein du domaine d'application du schéma de l'URI et de l'autorité de nommage (s'il en est). Le composant query est indiqué par le premier caractère point d'interrogation ("?") et se termine par un caractère dièse ("#") ou par la fin de l'URI.

```
query       = *( pchar / "/" / "?" )
```

Les caractères barre oblique ("/") et point d'interrogation ("?") peuvent représenter des données au sein du composant query. Il faut faire attention au fait que certaines mises en œuvre anciennes et erronées ne peuvent pas traiter correctement ces données lorsqu'elles sont utilisées comme URI de base pour des références croisées (paragraphe 5.1), apparemment parce qu'elles ne savent pas distinguer les données de query des données de path lorsqu'elles recherchent les séparateurs hiérarchiques. Cependant, comme les composants query sont souvent utilisés pour porter les informations d'identification de la forme de paires "key=value" (*clé* = *valeur*) et qu'une valeur fréquemment utilisée est une référence à un autre URI, il est parfois de meilleure utilisation d'éviter de coder en pourcentage ces caractères.

3.5. Fragment

Le composant identifiant de fragment d'un URI permet l'identification indirecte d'une ressource secondaire en référence à une ressource primaire et des informations d'identification supplémentaires. La ressource secondaire identifiée peut être une portion ou sous-ensemble de la ressource primaire, des éléments de représentations de la ressource primaire, ou d'autres ressources définies ou décrites par ces représentations. Un composant identifiant de fragment est indiqué par la présence d'un caractère dièse ("#") et terminé par la fin de l'URI.

```
fragment    = *( pchar / "/" / "?" )
```

La sémantique d'un identifiant de fragment est définie par l'ensemble des représentations qui peuvent résulter d'une action de restitution sur la ressource primaire. Le format et la résolution du fragment dépendent donc du type de support [RFC2046] d'une restitution de représentation potentielle, même si une telle restitution n'est effectuée que lorsque l'URI est dé référencé. S'il n'existe pas de telle représentation, la sémantique du fragment est considérée comme inconnue et reste sans conséquence. La sémantique d'identifiant de fragment est indépendante du schéma d'URI et ne peut donc être redéfinie par les spécifications du schéma.

Chaque type de support peut définir ses propres restrictions ou structures dans la syntaxe d'identifiant de fragment en spécifiant différents types de sous-ensembles, vues, ou références externes identifiables comme ressources secondaires par ce type de support. Si la ressource primaire a des représentations multiples, comme c'est souvent le cas de ressources dont la représentation est choisie sur la base d'attributs de la demande de restitution (a.k.a., négociation du contenu), tout ce qui est identifié par le fragment devrait alors être cohérent à travers toutes ces représentations. Chaque représentation devrait définir le fragment de façon qu'il corresponde à la même ressource secondaire, indépendamment de la façon dont il est représenté, ou laisser le fragment indéfini (c'est-à-dire, non trouvé). Comme avec tout URI, l'utilisation d'un composant d'identifiant de fragment n'implique pas qu'une action de restitution va avoir lieu. Un URI avec un identifiant de fragment peut être utilisé pour se référer à la ressource secondaire sans impliquer que la ressource primaire soit accessible ou qu'on y accèdera jamais.

Les identifiants de fragment jouent un rôle spécial dans les systèmes de restitution d'information en tant que forme de base de référencement indirect côté client, permettant à un auteur d'identifier de façon spécifique des aspects d'une ressource existante qui ne sont qu'indirectement fournis par le propriétaire de la ressource. Comme tel, l'identifiant de fragment n'est pas utilisé dans le traitement spécifique du schéma d'un URI ; au lieu de cela, l'identifiant de fragment est séparé du reste de l'URI avant un dé référencement, et donc, les informations d'identification au sein du fragment lui-même ne sont dé référencées que par l'agent utilisateur, sans considération du schéma d'URI. Bien que ce traitement séparé soit souvent perçu comme une perte d'information, particulièrement pour un réacheminement efficace des références lorsque les ressources sont déplacées, il sert aussi à empêcher les fournisseurs d'informations de refuser aux auteurs de références le

droit de se référer sélectivement aux informations à l'intérieur d'une ressource. Le référencement indirect donne aussi une souplesse et une capacité d'extension supplémentaires aux systèmes qui utilisent les URI, car les nouveaux types de support sont plus faciles à définir et développer que les nouveaux schémas d'identification.

Les caractères barre oblique ("/") et point d'interrogation ("?") sont permis pour représenter des données au sein de l'identifiant de fragment. Il faut faire attention au fait que certaines mises en œuvre anciennes et erronées peuvent ne pas traiter correctement ces données lorsqu'elles sont utilisées comme URI de base pour des références croisées (paragraphe 5.1).

4. Utilisation

Lorsque des applications font référence à un URI, elles n'utilisent pas toujours la forme complète de référence définie par la règle de la syntaxe "URI". Pour économiser l'espace et tirer parti de la localisation hiérarchique, de nombreux éléments de protocole Internet et formats de type de support permettent une abréviation de l'URI, alors que d'autres restreignent la syntaxe à une forme particulière d'URI. La présente spécification définit les formes les plus courantes de syntaxe de référence car elles ont une interaction avec les concepts de la syntaxe générique, qui requiert un algorithme d'analyse grammaticale uniforme afin d'être interprétée de façon cohérente.

4.1 Référence d'URI

URI-reference sert à noter l'utilisation la plus courante d'un identifiant de ressource.

```
URI-reference = URI / relative-ref
```

Une URI-reference est soit un URI soit une référence croisée. Si le préfixe de URI-reference ne correspond pas à la syntaxe d'un schéma suivi par ses deux points séparateurs, URI-reference est une référence croisée.

URI-reference est analysé le premier des cinq composants d'URI, afin de déterminer quels sont les composants présents et si la référence est croisée. Chaque composant est alors analysé dans ses sous-parties et leur validité. L'ABNF de URI-reference, conjointement avec la règle de résolution d'ambiguïtés "le premier qui correspond l'emporte", est suffisant pour définir un analyseur de validation pour la syntaxe générique. Les lecteurs familiers avec les expressions régulières se reporteront à l'Appendice B pour un exemple d'analyseur non validant de URI-reference qui accepte toute chaîne donnée et extrait les composants de l'URI.

4.2. Références croisées

Une référence croisée tire parti de la syntaxe hiérarchique (paragraphe 1.2.3) pour exprimer une référence d'URI relative à un espace de nom d'un autre URI hiérarchique.

```
relative-ref  = relative-part [ "?" query ] [ "#" fragment ]

relative-part = "//" authority path-abempty
              / path-absolute
              / path-noscheme
              / path-empty
```

L'URI auquel se réfère une référence croisée, appelé aussi URI cible, est obtenu en appliquant l'algorithme de résolution de référence de la Section 5.

Une référence croisée qui commence par deux caractères barre oblique est appelée une référence de chemin réseau ; de telles références sont rarement utilisées. Une référence croisée qui commence par un seul caractère barre oblique s'appelle une référence de chemin absolu. Une référence croisée qui ne commence pas par un caractère barre oblique est appelée une référence de chemin relatif.

Un segment path qui contient un caractère deux points (par exemple, "this:that") ne peut être utilisé comme premier segment d'une référence de chemin relatif, car il serait confondu avec un nom de schéma. Un tel segment doit être précédé d'un segment point (par exemple, "./this:that") pour faire une référence de chemin relatif.

4.3. Absolute URI

Certains éléments de protocole ne permettent que la forme absolue d'un URI, sans identifiant de fragment. Par exemple, définir un URI de base qui sera utilisé ultérieurement par des références croisées appelle une règle de syntaxe de absolute-URI qui n'admet pas de fragment.

```
absolute-URI  = scheme ":" hier-part [ "?" query ]
```

Les spécifications de schéma d'URI doivent définir leur propre syntaxe de sorte que toute chaîne correspondant à leur syntaxe spécifique de schéma corresponde aussi à la grammaire <absolute-URI>. Les spécifications de schéma ne définiront pas de syntaxe ou d'utilisation d'identifiant de fragment, indépendamment de son applicabilité à des ressources identifiables via ce schéma, car l'identification de fragment est orthogonale à la définition de schéma. Cependant, les spécifications de schéma sont invitées à inclure une large gamme d'exemples, y compris des exemples montrant l'utilisation des URI du schéma avec des identifiants de fragment lorsqu'une telle utilisation est appropriée.

4.4. Référence Same-Document

Lorsqu'une référence d'URI est faite à un URI qui est, à part son composant fragment (s'il en est), identique à l'URI de base (paragraphe 5.1), cette référence est appelée référence "same-document" (*même document*). Les exemples les plus fréquents de références même-document sont des références croisées qui sont vides ou n'incluent que le signe séparateur dièse ("#") suivi par un identifiant de fragment.
Lorsqu'une référence même-document est déréférencée pour une action de restitution, la cible de cette référence est définie comme étant dans la même entité (représentation, document, ou message) que la référence ; un dé référencement ne devrait donc pas avoir une nouvelle action de restitution pour résultat.

La normalisation de l'URI de base et de cible avant leur comparaison, comme décrit aux paragraphes 6.2.2 et 6.2.3, est admise mais rarement effectuée en pratique. La normalisation peut augmenter l'ensemble de références même-document, ce qui peut être profitable pour certaines applications à mémoire cache. Cela étant, les auteurs de références ne devraient pas supposer qu'une référence d'URI légèrement différente, bien qu'équivalente, sera (ou ne sera pas) interprétée comme une référence même-document par une application donnée.

4.5. Référence de suffixe

La syntaxe d'URI est conçue pour référencer de façon non ambiguë les ressources et extensions via un schéma d'URI. Cependant, comme l'identification et l'utilisation d'URI sont devenues très courantes, les supports traditionnels (télévision, radio, journaux, panneaux d'affichage, etc.) utilisent de plus en plus comme référence un suffixe de l'URI, comportant seulement les portions authority et path de l'URI, comme www.w3.org/Addressing/ ou simplement un nom enregistré DNS autonome. De telles références sont d'abord destinées à être interprétées par des humains plutôt que par des machines, avec l'hypothèse que les processus de recherche fondés sur le contexte sont suffisants pour compléter l'URI (par exemple, la plupart des noms enregistrés commençant par "www" ont vraisemblablement un préfixe d'URI de "http://"). Bien qu'il n'y ait pas d'ensemble normalisé des processus de recherche pour résoudre les ambiguïtés d'un suffixe d'URI, de nombreuses mises en œuvre client leur permettent d'être entrés par l'utilisateur et résolus par un processus heuristique.

Bien que cette façon d'utiliser les références de suffixe soit courante, elle devrait être évitée chaque fois que possible et ne devrait jamais être utilisée dans les situations où on attend des références à long terme. Les processus de recherche mentionnés ci-dessus vont évoluer avec le temps, et particulièrement avec la popularisation d'un nouveau schéma d'URI, et ils sont souvent incorrects lorsque utilisés hors contexte. De plus, ils peuvent poser des problèmes de sécurité comme ceux décrits dans la [RFC1535].

Comme un suffixe d'URI a la même syntaxe qu'une référence de chemin relatif, une référence de suffixe ne peut être utilisée dans des contextes où on attend une référence croisée. Il en résulte que les références de suffixe sont limitées aux endroits où il n'y a pas d'URI de base défini, tels que boîte de dialogue et publicités hors ligne.

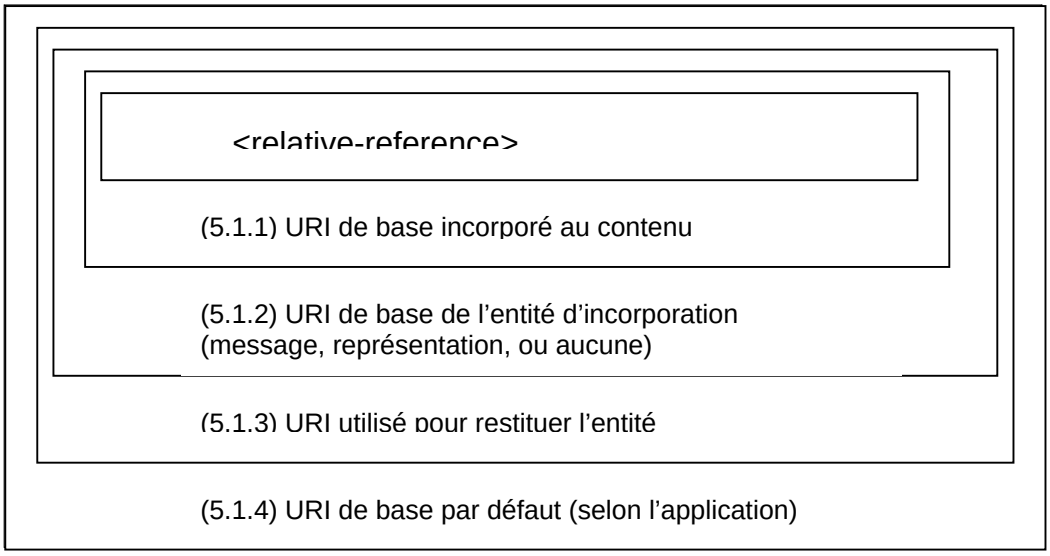
5. Résolution de référence

La présente section définit le processus de résolution d'une référence d'URI au sein d'un contexte qui admet les références croisées de sorte qu'il en résulte une chaîne satisfaisant à la règle syntaxique <URI> de la Section 3.

5.1. Etablissement d'un URI de base

Le terme "relative" (*croisée*) implique qu'il existe un "URI de base" par rapport auquel la référence croisée s'applique. En dehors des références fragment-seul (paragraphe 4.4), les références croisées ne sont utilisables que lorsque un URI de base est connu. Un URI de base doit être établi par l'analyseur grammatical avant d'analyser les références d'URI qui pourraient être croisées. Un URI de base doit se conformer à la règle syntaxique <absolute-URI> (paragraphe 4.3). Si l'URI de base est obtenu à partir d'une référence d'URI, cette référence doit alors être convertie en forme absolue et dépouillée de tout composant de fragment avant d'être utilisé comme URI de base.

L'URI de base d'une référence peut être établi d'une des quatre façons exposées ci-dessous par ordre de préséance. L'ordre de préséance peut être vu en termes de couches, où l'URI de base défini comme le plus interne a la préséance la plus élevée. Ceci peut être visualisé de la façon suivante:



5.1.1. URI de base incorporé dans le contenu

Dans certains types de support, un URI de base pour des références croisées peut être incorporé dans le contenu lui-même de sorte qu'il puisse être obtenu directement par un analyseur. Ceci peut être utile pour des documents descriptifs, tels que des tableaux de contenu, qui peuvent être transmis à d'autres au moyen de protocoles autres que ceux de leur contexte de restitution habituel (par exemple, email ou USENET news).

Il est en dehors du domaine d'application de la présente spécification de spécifier, pour chaque type de support, comment peut être incorporé un URI de base. La syntaxe appropriée, lorsqu'elle est disponible, est décrite par la spécification de format de données associée à chaque type de support.

5.1.2. URI de base tiré de l'entité incorporatrice

Si aucun URI de base n'est incorporé, l'URI de base est défini par le contexte de restitution de la représentation. Pour un document qui est inclus dans une autre entité, tel qu'un message ou une archive, le contexte de restitution est cette entité. Et donc, l'URI de base par défaut d'une représentation est l'URI de base de l'entité dans laquelle la représentation est incorporée.

Un mécanisme d'incorporation d'un URI de base au sein des types de contenants MIME (par exemple, le message et des types multiparties) est défini par MHTML [RFC2557]. Les protocoles qui n'utilisent pas la syntaxe d'en-tête de message MIME, mais admettent que certaines formes de métadonnées étiquetées soient inclues dans les messages, peuvent définir leur propre syntaxe de définition d'URI de base en tant que partie d'un message.

5.1.3. URI de base à partir de l'URI de restitution

Si aucun URI de base n'est inclus et que la représentation n'est pas incorporée au sein de quelque autre entité, lorsqu'un URI a été utilisé pour restituer la représentation, cet URI doit être considéré comme URI de base. Noter que si la restitution a été le résultat d'une demande redirigée, le dernier URI utilisé (c'est-à-dire, l'URI qui apparaît dans la restitution réelle de la représentation) est l'URI de base.

5.1.4. URI de base par défaut

Si aucune des conditions décrites ci-dessus ne s'applique, l'URI de base est alors défini par le contexte de l'application. Cette définition dépendant nécessairement de l'application, l'échec à définir un URI de base par l'utilisation de l'une des autres méthodes peut aboutir à donner au même contenu une interprétation différente selon les types d'applications.

L'expéditeur d'une représentation contenant des références croisées est responsable de l'établissement d'un URI de base pour ces références. A côté des références fragment seul, les références croisées ne peuvent être utilisées avec fiabilité que dans les situations où l'URI de base est bien défini.

5.2 Résolution de référence croisée

Le présent paragraphe décrit un algorithme de conversion de référence d'URI qui pourrait être croisée, avec un URI de base donné selon les composants analysés grammaticalement de la cible de la référence. Les composants peuvent alors être recomposés, comme décrit au paragraphe 5.3, pour former l'URI cible. Cet algorithme donne les résultats définitifs qui peuvent être utilisés pour vérifier le résultat des autres mises en œuvre. Les applications peuvent mettre en œuvre la résolution de références croisées en utilisant d'autres algorithmes, pourvu que le résultat corresponde à ce qui aurait été donné par celui-ci.

5.2.1. Pré-analyse de l'URI de base

L'URI de base (Base) est établi conformément à la procédure du paragraphe 5.1 et analysé dans les cinq composants principaux décrits à la Section 3. Noter que seule la présence du composant schéma est nécessaire dans un URI de base ; les autres composants peuvent être vides ou indéfinis. Un composant est indéfini si son délimiteur associé n'apparaît pas dans la référence d'URI ; le composant path n'est jamais indéfini, bien qu'il puisse être vide.

La normalisation de l'URI de base, décrite aux paragraphes 6.2.2 et 6.2.3, est facultative. Une référence d'URI doit être transformée en son URI cible avant qu'il puisse être normalisé.

5.2.2. Transformation de références

Pour chaque référence d'URI (R), le pseudocode suivant décrit un algorithme de transformation de R en son URI cible (T) :

```
-- La référence d'URI est analysée dans les cinq composants d'URI
-- (R.scheme, R.authority, R.path, R.query, R.fragment) = parse(R);
-- Un analyseur peu rigoureux peut ignorer un schéma dans la référence
-- si elle est identique au schéma de l'URI de base.
--si ((non strict) et (R.scheme == Base.scheme)) alors (R.scheme)indéfini;
endif;
si défini(R.scheme) alors
   T.scheme    = R.scheme;
   T.authority = R.authority;
   T.path      = retirer_dot_segments(R.path);
   T.query     = R.query;
autrement
   si défini(R.authority) alors
      T.authority = R.authority;
      T.path      = retirer_dot_segments(R.path);
      T.query     = R.query;
   autrement
      si (R.path == "") alors
         T.path = Base.path;
```

```
        si défini(R.query) alors
          T.query = R.query;
        autrement
          T.query = Base.query;
        endif;
      autrement
        si (R.path commence-par "/") alors
          T.path = retirer_dot_segments(R.path);
        autrement
          T.path = fusionner(Base.path, R.path);
          T.path = retirer_dot_segments(T.path);
        endif;
        T.query = R.query;
      endif;
      T.authority = Base.authority;
    endif;
    T.scheme = Base.scheme;
  endif;

  T.fragment = R.fragment;
```

5.2.3. Fusion de chemins

Le pseudocode ci-dessus se réfère à une instruction "fusionner" (*merge*) pour fusionner une référence de chemin relatif avec le chemin de l'URI de base. Ceci est réalisé comme suit:

- Si l'URI de base a un composant d'autorité défini et un chemin vide, il faut alors retourner une chaîne consistant en "/" enchaîné avec le chemin de référence, autrement
- Retourner une chaîne comportant le composant de chemin de la référence ajouté à tous les segments, sauf le dernier du chemin de l'URI de base (c'est-à-dire, excluant tout caractère après la "/" la plus à droite dans le chemin de l'URI de base, ou excluant la totalité du chemin de l'URI de base si il ne contient pas de caractère "/").

5.2.4. Retirer les segments point

Le pseudocode se réfère aussi à une procédure "remove_dot_segments" pour l'interprétation et le retrait des segments de chemin complet de "." et ".." spéciaux d'un chemin référencé. Cela est fait après l'extraction du chemin de la référence, que le chemin soit ou non relatif, afin de retirer tout dot_segment invalide ou étranger avant de former l'URI de cible. Bien qu'il y ait de nombreuses façons de réaliser ce processus de retrait, on décrit ici une méthode simple qui utilise deux mémoires tampons de chaîne.

1. La mémoire tampon d'entrée est initialisée avec les composants de chemin qu'on vient d'ajouter et la mémoire tampon de sortie est initialisée sur la chaîne vide.

2. Alors que la mémoire tampon d'entrée n'est pas vide, boucler comme suit :

   A. Si la mémoire tampon d'entrée commence par un préfixe de "../" ou "./", retirer ce préfixe de la mémoire d'entrée ; autrement,

   B. Si la mémoire d'entrée commence par un préfixe de "/./" o "/.", où "." est un segment de chemin complet, remplacer ce préfixe par "/" dans la mémoire d'entrée ; autrement,

   C. Si la mémoire d'entrée commence par un préfixe de "/../" ou "/..", où ".." est un segment de chemin complet, remplacer ce préfixe par "/" dans la mémoire d'entrée et retirer le dernier segment et ses "/" précédentes (s'il en est) de la mémoire de sortie ; autrement,

   D. Si la mémoire d'entrée ne comporte que des "." ou "..", les retirer de la mémoire d'entrée ; autrement,

   E. Déplacer le premier segment de chemin de la mémoire d'entrée à la fin de la mémoire de sortie, y compris le caractère "/" initial (s'il en est) et tout caractère suivant jusqu'à, non inclus, le prochain caractère "/" ou la fin de la mémoire d'entrée.

3. Finalement, la mémoire de sortie est retournée comme résultat de remove_dot_segments.

Noter que dans les références d'URI, les dot-segments sont destinés à être utilisés pour exprimer un identifiant par rapport à la hiérarchie des noms dans l'URI de base. L'algorithme remove_dot_segments respecte cette hiérarchie en retirant les dot-segments surabondants plutôt que de les traiter comme une erreur ou les laisser au risque d'une mauvaise interprétation par les mises en œuvre de dé référencement.

Ce qui suit illustre comment sont appliquées les étapes précédentes par deux exemples de chemins fusionnés, qui montrent l'état des deux mémoires tampon après chaque étape.

ETAPE	MEMOIRE DE SORTIE	MEMOIRE D'ENTREE
1 :		/a/b/c/./../../g
2E:	/a	/b/c/./../../g
2E:	/a/b	/c/./../../g
2E:	/a/b/c	/./../../g
2B	/a/b/c	/../../g
2C:	/a/b	/../g
2C:	/a	/g
2E:	/a/g	

ETAPE	MEMOIRE DE SORTIE	MEMOIRE D'ENTREE
1 :		mid/content=5/../6
2E:	mid	/content=5/../6
2E:	mid/content=5	/../6
2C:	mid	/6
2E:	mid/6	

Certaines applications peuvent trouver plus efficace de mettre en œuvre l'algorithme remove_dot_segments en utilisant deux piles de segments plutôt que des chaînes.

Note : Il faut faire attention au fait que certaines mises en œuvre anciennes et erronées échouent à séparer un composant d'interrogation de référence de son composant de chemin avant de fusionner les chemins de base et de référence, ce qui donne un échec d'interfonctionnement si le composant d'interrogation contient la chaîne "/../" ou "/./".

5.3. Recomposition de composant

Les composants d'URI analysés peuvent être recomposés pour obtenir la chaîne de référence d'URI correspondante. En utilisant un pseudocode, cela donnerait :

```
result = ""

si defined(scheme) alors
   ajouter scheme au résultat;
   ajouter ":" au résultat;
endif;

si defined(authority) alors
   ajouter "//" au résultat;
   ajouter authority au résultat;
endif;

ajouter path au résultat;

si defined(query) alors
   ajouter "?" au résultat;
   ajouter query au résultat;
endif;

si defined(fragment) alors
   ajouter "#" au résultat;
   ajouter fragment au résultat;
endif;
```

retourner résultat;

Noter que nous avons fait attention à préserver la distinction entre un composant indéfini, signifiant que son séparateur ne figure pas dans la référence, et un composant vide, signifiant que le séparateur est présent et est immédiatement suivi du séparateur du composant suivant ou de la fin de la référence.

5.4. Exemples de résolution de références

Au sein d'une représentation d'un URI de base bien défini de http://a/b/c/d;p?q

Une référence croisée est transformée en son URI cible comme suit.

5.4.1. Exemples normaux

```
"g:h"           =  "g:h"
"g"             =  "http://a/b/c/g"
"./g"           =  "http://a/b/c/g"
"g/"            =  "http://a/b/c/g/"
"/g"            =  "http://a/g"
"//g"           =  "http://g"
"?y"            =  "http://a/b/c/d;p?y"
"g?y"           =  "http://a/b/c/g?y"
"#s"            =  "http://a/b/c/d;p?q#s"
"g#s"           =  "http://a/b/c/g#s"
"g?y#s"         =  "http://a/b/c/g?y#s"
";x"            =  "http://a/b/c/;x"
"g;x"           =  "http://a/b/c/g;x"
"g;x?y#s"       =  "http://a/b/c/g;x?y#s"
""              =  "http://a/b/c/d;p?q"
"."             =  "http://a/b/c/"
"./"            =  "http://a/b/c/"
".."            =  "http://a/b/"
"../"           =  "http://a/b/"
"../g"          =  "http://a/b/g"
"../.."         =  "http://a/"
"../../"        =  "http://a/"
"../../g"       =  "http://a/g"
```

5.4.2 Exemples anormaux

Bien que les exemples anormaux suivants aient peu de chances de survenir en pratique, tous les analyseurs d'URI devraient être capables de les résoudre de façon cohérente. Chaque exemple utilise la même base que celle ci-dessus.

Les analyseurs doivent être attentifs au traitement des cas où il y a plus de segments ".." dans une référence de chemin relatif qu'il n'y a de niveaux hiérarchiques dans le chemin de l'URI de base. Noter que la syntaxe ".." ne peut pas être utilisée pour changer le composant authority d'un URI.

```
"../../../g"    =  "http://a/g"
"../../../../g" =  "http://a/g"
```

De même, les analyseurs doivent retirer les segments point "." et ".." lorsqu'ils sont les composants complets d'un chemin, mais pas lorsqu'ils sont seulement une partie d'un segment.

```
"/./g"          =  "http://a/g"
"/../g"         =  "http://a/g"
"g."            =  "http://a/b/c/g."
".g"            =  "http://a/b/c/.g"
"g.."           =  "http://a/b/c/g.."
"..g"           =  "http://a/b/c/..g"
```

Moins vraisemblables sont les cas où la référence croisée utilise des formes non nécessaires ou absurdes de segments de chemin complets de "." et "..".

```
"./../g"       =  "http://a/b/g"
"./g/."        =  "http://a/b/c/g/"
"g/./h"        =  "http://a/b/c/g/h"
"g/../h"       =  "http://a/b/c/h"
"g;x=1/./y"    =  "http://a/b/c/g;x=1/y"
"g;x=1/../y"   =  "http://a/b/c/y"
```

Certaines applications ne réussissent pas à séparer les composants d'interrogation et/ou de fragment de la référence du composant de chemin avant de le fusionner avec le chemin de base et de retirer les segments point. Cette erreur se remarque rarement, car l'utilisation normale d'un fragment n'inclut jamais le caractère de hiérarchie ("/") et le composant query n'est normalement pas utilisé au sein de références croisées.

```
"g?y/./x"      =  "http://a/b/c/g?y/./x"
"g?y/../x"     =  "http://a/b/c/g?y/../x"
"g#s/./x"      =  "http://a/b/c/g#s/./x"
"g#s/../x"     =  "http://a/b/c/g#s/../x"
```

Certains analyseurs permettent que le nom de schéma soit présent dans une référence croisée si il est le même que celui du schéma de l'URI de base. Ceci est considéré comme une échappatoire dans les spécifications précédentes d'URI partiel [RFC1630]. Son utilisation devrait être évitée mais elle est permise pour la compatibilité amont.

```
"http:g"       =  "http:g"        ; pour les analyseurs stricts
               /  "http://a/b/c/g" ; pour la compatibilité amont
```

6. Normalisation et comparaison

Une des opérations les plus courantes sur les URI est la simple comparaison : déterminer si deux URI sont équivalents sans utiliser les URI pour accéder à leurs ressources respectives. Une comparaison est effectuée chaque fois qu'on accède à une mémoire cache de réponse, qu'un navigateur vérifie son historique pour valider un lien, ou qu'un analyseur XML traite des étiquettes dans un espace de nom. Les spiders et moteurs d'indexation utilisent souvent une normalisation extensive avant de comparer les URI pour élaguer l'espace de recherche ou pour réduire les duplications d'actions de demande et de stockage de réponses.

La comparaison d'URI est effectuée dans des buts précis. Les protocoles ou mises en œuvre qui comparent les URI pour différents objets font souvent l'objet de divergences de conception quant à l'évaluation de la quantité d'efforts qui doivent être déployés pour réduire les noms d'emprunt d'identifiants. La présente section décrit diverses méthodes qui peuvent être utilisées pour comparer les URI, les échanges entre eux, et les types d'applications qui pourraient les utiliser.

6.1. Equivalence

Comme les URI existent pour identifier les ressources, on peut présumer qu'ils devraient être considérés comme équivalents lorsqu'ils identifient la même ressource. Cependant, cette définition d'équivalence n'est pas d'un grand usage pratique, car une mise en œuvre n'a aucun moyen de comparer deux ressources si elle n'a pas une connaissance et un contrôle complet sur elles. Pour cette raison, la détermination de l'équivalence ou de la différence des URI se fonde sur la comparaison des chaînes, qui peut être améliorée par la référence à des règles supplémentaires fournies par les définitions de schéma d'URI. Les termes "différent" et "équivalent" sont utilisés pour décrire les résultats possibles de telles comparaisons, mais il y a de nombreuses versions dépendantes de l'application de la notion d'équivalence.

Même s'il est possible de déterminer que deux URI sont équivalents, la comparaison d'URI n'est pas suffisante pour déterminer si deux URI identifient des ressources différentes. Par exemple, le propriétaire de deux noms de domaine différents peut décider de desservir la même ressource à partir des deux, ce qui donnera deux URI différents. Donc, les méthodes de comparaison sont conçues pour minimiser les faux négatifs tout en évitant strictement les faux positifs.

Dans les essais d'équivalence, les applications ne devraient pas comparer directement les références croisées ; les références devraient être converties en leur URI cible respectif avant la comparaison. Lorsque les URI sont comparés pour choisir (ou éviter) une action de réseau, comme la restitution d'une représentation, les composants de fragment (s'il en est) devraient être exclus de la comparaison.

6.2. Echelle de comparaison

Diverses méthodes sont utilisées en pratique pour vérifier l'équivalence d'URI. Ces méthodes se classent en distinguant selon la quantité de traitement nécessaire et le degré de réduction de la probabilité de faux négatif. Comme noté ci-dessus, les faux négatifs ne peuvent être éliminés. En pratique, leur probabilité peut être réduite, mais cette réduction exige plus de traitement et n'est pas rentable pour toutes les applications.

Si cette gamme de pratiques de comparaison est considérée comme une échelle, l'examen suivant va remonter l'échelle, en commençant pas les pratiques qui sont peu coûteuses mais ont une chance relativement grande de produire des faux négatifs, et en remontant jusqu'à celles qui ont un coût de traitement plus élevé et un moindre risque de faux négatifs.

6.2.1. Comparaison de chaîne simple

Si deux URI, considérés comme chaînes de caractères, sont identiques, on peut conclure en toute sécurité qu'ils sont équivalents. Ce type d'essai d'équivalence a un très faible coût de traitement et est d'un large usage dans des applications très diverses, en particulier dans le domaine de l'analyse grammaticale.

Les chaînes d'essai d'équivalence requièrent quelques précautions de base. Cette procédure est souvent désignée sous le nom de comparaison "bit à bit" ou "octet par octet", ce qui peut induire en erreur. L'essai des chaînes pour vérifier leur égalité se fonde normalement sur la comparaison paire par paire des caractères qui composent les chaînes, en commençant par la première et en continuant jusqu'à épuisement des deux chaînes et que tous les caractères aient été trouvés égaux, jusqu'à ce qu'une paire de caractères se révèle inégale ou qu'une des chaînes arrive à épuisement avant l'autre.
Cette comparaison de caractères exige que chaque paire de caractères puisse être mise dans une forme comparable. Par exemple, si on a un URI stocké sur une matrice binaire en codage EBCDIC et que le second est un objet Java String (UTF-16), les comparaisons bit à bit appliquées de façon ingénue produiront des erreurs. Il vaut mieux parler d'égalité caractère par caractère plutôt que bit à bit ou octet par octet. Pour parler de façon concrète, les comparaisons caractère par caractère devraient être faites point de code par point de code après conversion à un codage de caractères commun.

Les faux négatifs sont causés par la production et l'utilisation de noms d'emprunt d'URI. Les noms d'emprunt non nécessaires peuvent être réduits, indépendamment de la méthode de comparaison, en fournissant de façon cohérente les références d'URI sous une forme normalisée (c'est-à-dire, une forme identique à ce qui serait produit après application de la normalisation, comme décrit ci-dessous).

Les protocoles et formats de données limitent souvent certaines comparaisons d'URI à une simple comparaison de chaînes, fondée sur la théorie que les gens et les mises en oeuvre vont, dans leur propre intérêt, être cohérents dans la fourniture des références d'URI, ou au minimum, assez cohérents pour annihiler toute efficacité qui pourrait être obtenue d'une normalisation ultérieure.

6.2.2. Normalisation fondée sur la syntaxe

Les mises en œuvre peuvent utiliser une logique fondée sur les définitions données par la présente spécification pour réduire la probabilité de faux négatifs. Ce traitement est d'un coût modérément plus élevé que la comparaison de chaînes caractère par caractère. Par exemple, une application utilisant cette approche pourrait raisonnablement considérer les deux URI suivants comme équivalents :

example://a/b/c/%7Bfoo%7D
eXAMPLE://a/./b/../b/%63/%7bfoo%7d

Les agents d'usager du Web, comme les navigateurs, appliquent normalement ce type de normalisation d'URI pour déterminer si une réponse est disponible en mémoire cache. La normalisation fondée sur la syntaxe inclut des techniques comme la normalisation de casse, la normalisation de codage en pourcentage, et le retrait des segments point.

6.2.2.1. Normalisation de casse

Pour tous les URI, les chiffres hexadécimaux au sein d'un triplet codé en pourcentage (par exemple, "%3a" contre "%3A") sont insensibles à la casse et devraient donc être normalisés à l'utilisation des majuscules pour les chiffres A à F.

Lorsqu'un URI utilise des composants de la syntaxe générique, les règles d'équivalence syntaxique du composant s'appliquent toujours ; à savoir, que le schéma et l'hôte sont insensibles à la casse et donc devraient être normalisés en minuscules. Par exemple, l'URI <HTTP://www.EXAMPLE.com/> est équivalent à <http://www.example.com/>. Les autres composants de syntaxe générique sont supposés être sensibles à la casse à moins qu'il n'en soit spécifiquement décidé autrement par le schéma (voir le paragraphe 6.2.3).

6.2.2.2. Normalisation de codage en pourcentage

Le mécanisme du codage en pourcentage (paragraphe 2.1) est une source fréquente de variation entre des URI identiques par ailleurs. En plus de la question de la normalisation de la casse mentionnée ci-dessus, certains producteurs d'URI codent en pourcentage des octets qui ne le requièrent pas, provoquant l'équivalence d'URI avec des contreparties non codées. Ces URI devraient être normalisés en décodant tout octet codé en pourcentage qui correspond à un caractère non réservé, comme décrit au paragraphe 2.3.

6.2.2.3 Normalisation du segment chemin

Les segments chemin complets "." et ".." sont destinés à être uniquement utilisés dans des références croisées (Section 4.1) et sont retirés dans le cours du processus de résolution de la référence (paragraphe 5.2). Cependant, certaines mises en œuvre développées de façon incorrecte supposent que la résolution de références n'est pas nécessaire lorsque la référence est déjà un URI et omettent donc de retirer les segments point lorsqu'ils surviennent dans des chemins non relatifs. Les normalisateurs d'URI devraient retirer les segments point en appliquant l'algorithme remove_dot_segments au chemin, comme décrit au paragraphe 5.2.4.

6.2.3 Normalisation fondée sur le schéma

La syntaxe et la sémantique des URI varie d'un schéma à l'autre, comme décrit par la spécification de définition de chaque schéma. Les mises en œuvre peuvent utiliser des règles spécifiques du schéma, pour un coût de traitement supérieur, pour réduire la probabilité de faux négatifs. Par exemple, puisque le schéma "http" utilise un composant authority, a un port par défaut de "80", et définit un chemin vide comme équivalent à "/", les quatre URI suivants sont équivalents :

```
http://example.com
http://example.com/
http://example.com:/
http://example.com:80/
```

En général, un URI qui utilise la syntaxe générique pour authority avec un chemin vide devrait être normalisé en un chemin de "/". De même, un ":port" explicite, pour lequel le port est vide ou la valeur par défaut pour le schéma, est équivalent à un URI où le port et son délimiteur ":" sont élidés et devraient donc être retirés par la normalisation fondée sur le schéma. Par exemple, le second URI ci-dessus est la forme normale pour le schéma "http".

Un autre cas où la normalisation varie selon le schéma est le traitement d'un composant authority vide ou d'un sous-composant hôte vide. Pour de nombreuses spécifications de schéma, une autorité ou hôte vide est considéré comme une erreur ; pour d'autres, c'est considéré comme équivalent à "localhost" ou à l'hôte de l'utilisateur final. Lorsqu'un schéma définit une valeur par défaut pour l'autorité et qu'une référence d'URI à cette valeur par défaut est souhaitée, la référence devrait être normalisée à une autorité vide au nom de l'uniformité, de la concision, et de l'internationalisation. Si cependant, les sous-composants userinfo ou port sont non vides, l'hôte devrait être donné de façon explicite même s'il correspond à la valeur par défaut.

La normalisation ne devrait pas retirer les délimiteurs lorsque leur composant associé est vide, à moins que cela ne soit autorisé par la spécification de schéma. Par exemple, l'URI "http://example.com/?" ne peut être supposé équivalent à aucun des exemples ci-dessus. De même, la présence ou l'absence de délimiteurs dans un sous-composant userinfo est normalement significative pour son interprétation. Le composant fragment n'est soumis à aucune normalisation fondée sur le schéma ; et donc, deux URI qui diffèrent seulement par le suffixe "#" sont considérés comme différents quel que soit leur schéma.

Certains schémas définissent des sous-composants supplémentaires qui consistent en données insensibles à la casse, donnant une licence implicite aux normalisateurs de convertir ces données à une casse commune (par exemple, tout en minuscule). Par exemple, les schémas d'URI qui définissent un sous-composant de path pour contenir un nom d'hôte Internet, tel que le schéma d'URI "mailto", amènent ce sous-composant à être insensible à la casse et donc sujet à la normalisation de casse (par exemple, "mailto:Joe@Example.COM" est équivalent à "mailto:Joe@example.com", même si la syntaxe générique considère le composant path comme sensible à la casse).

D'autres normalisations spécifiques du schéma sont possibles.

6.2.4. Normalisation fondée sur le protocole

Un effort substantiel de réduction de l'incidence des faux négatifs est souvent rentable pour les accélérateurs de web. Ils mettent donc en œuvre des techniques encore plus agressives pour la comparaison d'URI. Par exemple, si ils observent qu'un URI tel que http://example.com/data redirige sur un URI qui en diffère seulement par la barre oblique de queue http://example.com/data/ ils vont vraisemblablement considérer à l'avenir que les deux sont équivalents. Cette sorte de technique n'est appropriée que lorsque l'équivalence est clairement indiquée à la fois par le résultat de l'accès aux ressources et les conventions communes de l'algorithme de dé référencement de leur schéma (dans ce cas, l'utilisation de la redirection par les serveurs HTTP d'origine pour éviter les problèmes de références croisées).

7. Considérations de sécurité

En lui-même un URI ne constitue pas une menace pour la sécurité. Cependant, comme les URI sont souvent utilisés pour donner un ensemble compact d'instructions d'accès à des ressources réseau, il faut prendre garde à bien interpréter les données dans un URI, pour empêcher que ces données ne causent d'accès non intentionnel, et pour éviter d'inclure des données qui ne devraient pas être révélées en texte clair.

7.1. Fiabilité et cohérence

Il n'y a aucune garantie qu'une fois qu'un URI a été utilisé pour restituer des informations, les mêmes informations soient restituables à l'avenir par cet URI. Pas plus qu'il n'est garanti que les informations restituables via cet URI à l'avenir seront visiblement similaires à celles restituées dans le passé. La syntaxe d'URI ne crée pas de contraintes sur la façon dont un schéma ou une autorité donné répartit l'utilisation de son espace de nom ou en assure la maintenance dans le temps. De telles garanties ne peuvent être obtenues que de la ou des personnes qui contrôlent cet espace et la ressource en question. Un schéma d'URI spécifique peut définir une sémantique supplémentaire, telle que la persistance du nom, si cette sémantique est requise de toutes les autorités de nommage pour ce schéma.

7.2. Construction nuisible

Il est parfois possible de construire un URI de telle sorte qu'une tentative d'effectuer une opération apparemment anodine et sans danger, telle que la restitution d'une représentation, va en fait causer une opération à distance potentiellement dommageable. L'URI douteux est normalement construit en spécifiant un numéro de port autre que celui réservé pour le protocole réseau en question. Le client contacte contre son gré un site gérant un service de protocole différent, et les données au sein de l'URI contiennent des instructions qui, interprétées conformément à cet autre protocole, causent une opération inattendue. Un exemple fréquent d'un tel abus a été l'utilisation d'un schéma fondé sur le protocole avec un composant de port de "25", trompant ainsi le logiciel agent d'utilisateur en envoyant un message non intentionnel ou trompeur via un serveur SMTP.

Les applications devraient empêcher le dé référencement d'un URI qui spécifie un numéro de port TCP dans la gamme "de port bien connue" (0 à 1023) à moins que le protocole utilisé pour le dé référencement de cet URI ne soit compatible avec le protocole attendu sur ce port bien connu. Bien que l'IANA maintienne un registre des ports bien connus, les applications devraient faire de telles restrictions configurables par l'utilisateur pour éviter d'entraver le développement des nouveaux services.

Lorsqu'un URI contient des octets codés en pourcentage qui correspondent aux délimiteurs pour un protocole de résolution ou de dé référencement donné (par exemple, des caractères CR et LF pour le protocole

TELNET), ces codages en pourcentage ne doivent pas être décodés avant transmission à travers ce protocole. Le transfert de codages en pourcentage, qui peut violer le protocole, est moins dommageable que de permettre que des octets décodés soient interprétés comme des opérations ou paramètres supplémentaires, déclenchant peut-être une opération distante inattendue et pouvant être dommageable.

7.3. Transcodage d'extrémité arrière

Lorsqu'un URI est dé référencé, les données qu'il contient sont souvent analysées à la fois par l'agent d'utilisateur et par un ou plusieurs serveurs. En HTTP, par exemple, un agent d'utilisateur normal va analyser un URI selon ses cinq composants majeurs, accéder au serveur de l'autorité, et lui envoyer les données selon les composants authority, path, et query. Un serveur normal va prendre ces informations, analyser le chemin selon ses segments et la question en paires clé/valeur, puis invoquer les traitements spécifiques de la mise en œuvre pour répondre à la demande. Il en résulte qu'un souci de sécurité commun aux mises en œuvre de serveurs qui traitent un URI, comme un tout ou divisé en composants séparés, est une interprétation adéquate des octets de données représentés par les caractères et codages en pourcentage dans cet URI.

Les octets codés en pourcentage doivent être décodés en un lieu durant le processus de dé référencement. Les applications doivent séparer l' URI selon ses composants et sous-composants avant de décoder les octets, car autrement les octets décodés pourraient être pris pour des délimiteurs. Les vérifications de sécurité des données contenues dans un URI devraient être appliquées après le décodage des octets. Noter, cependant, que le codage en pourcentage "%00" (NUL) peut demander un traitement spécial et qu'il devrait être rejeté si l'application ne s'attend pas à recevoir des données brutes au sein d'un composant.

Une attention particulière devrait être apportée lorsque le processus d'interprétation du chemin de l'URI implique l'utilisation d'un système de fichiers d'extrémité arrière ou de fonctions système s'y rapportant. Les fichiers systèmes allouent normalement une signification opérationnelle aux caractères spéciaux, tels que les caractères "/", "\", ":", "[", et "]", et aux noms de dispositifs spéciaux tels que ".", "..", "...", "aux", "lpt", etc. Dans certains cas, le simple essai d'existence d'un tel nom va causer l'arrêt du système d'exploitation ou invoquer des appels sans relation avec le système, conduisant à des problèmes de sécurité significatifs de déni de service et de transfert de données involontaires. Il serait impossible à la présente spécification de faire la liste de tous les caractères et noms de dispositifs significatifs. Les auteurs de mises en œuvre devraient rechercher les noms et caractères réservés pour les types de dispositifs de stockage qui peuvent être reliés à leurs applications et restreindre en conséquence l'utilisation des données obtenues de composants d'URI.

7.4. Formats d'adresse IP rares

Bien que la syntaxe d'URI pour les adresses IPv4 ne permette que la forme décimale à point d'adresse littérale IPv4, de nombreuses mises en œuvre qui traitent des URI utilisent des routines système qui dépendent de la plate-forme, comme gethostbyname() et inet_aton(), pour traduire la chaîne littérale en adresse IP réelle. Malheureusement, de telles routines système permettent et traitent souvent un ensemble de formats beaucoup plus grands que celui décrit au paragraphe 3.2.2.

Par exemple, de nombreuses mises en œuvre permettent des formes à point de trois nombres, dans lesquels la dernière partie est interprétée comme une quantité de 16 bits et placée dans les deux octets les plus à droite de l'adresse réseau (par exemple, un réseau de classe B). De même, une forme à point de deux nombres signifie que la dernière partie est interprétée comme une quantité de 24 bits et placée dans les trois octets les plus à droite de l'adresse réseau (Classe A), et un seul nombre (sans point) est interprété comme une quantité de 2 bits et mémorisée directement dans l'adresse réseau. Pour ajouter à la confusion, certaines mises en œuvre permettent que chaque partie à point soit interprétée comme décimal, octal, ou hexadécimal, comme spécifié en langage C (c'est-à-dire, un début en 0x ou 0X implique l'hexadécimal ; un début en 0 implique l'octal ; autrement, le nombre est interprété comme décimal). Ces formats d'adresse IP additionnels ne sont pas admis dans la syntaxe d'URI à cause des différences entre les mises en œuvre de plates-formes. Cependant, elles peuvent devenir un problème de sécurité si une application essaye de filtrer l'accès à des ressources sur la base de l'adresse IP dans un format littéral de chaîne. Si ce filtrage est effectué, les chaînes littérales devraient être converties à la forme numérique et filtrées sur la base de la valeur numérique, et non sur un préfixe ou suffixe en forme de chaîne.

7.5. Informations sensibles

Les producteurs d'URI ne devraient pas fournir d'URI contenant un nom d'utilisateur ou un mot de passe destiné à rester secret. Les URI sont fréquemment affichés par les navigateurs, mémorisés en signets en

clair, et enregistrés par l'historique de l'agent d'utilisateur et les applications intermédiaires (mandataires). Un mot de passe qui apparaît dans le composant userinfo est déconseillé et devrait être considéré comme une erreur (ou simplement ignoré) excepté dans les rares cas où le paramètre 'password' (*mot de passe*) est destiné à être public.

7.6. Attaques sémantiques

Comme le sous-composant userinfo est rarement utilisé et apparaît avant l'hôte dans le composant authority, il peut être utilisé pour construire un URI destiné à tromper un utilisateur humain en apparaissant comme identifiant une autorité de nommage (de confiance) alors qu'en réalité il identifie une autorité différente cachée derrière un écran de fumée. Par exemple

ftp://cnn.example.com&story=breaking_news@10.0.0.1/top_story.htm

peut conduire un utilisateur humain à supposer que l'hôte est 'cnn.example.com', alors qu'il est en réalité '10.0.0.1'. Noter qu'un sous-composant userinfo trompeur pourrait être beaucoup plus long que l'exemple ci-dessus.

Un URI trompeur, tel que celui ci-dessus, est une attaque fondée sur les notions préconçues de l'utilisateur sur la signification d'un URI plutôt qu'une attaque contre le logiciel lui-même. Les agents d'utilisateurs peuvent être capables de réduire l'impact de telles attaques en distinguant entre les divers composants de l'URI lorsqu'ils sont clarifiés, comme en utilisant une couleur ou un ton différent pour rendre le userinfo, s'il en est un présent, bien que ce ne soit pas une panacée. On trouvera des informations complémentaires sur les attaques fondées sur la sémantique d'URI dans [Siedzik].

8. Considérations sur l'IANA

Les noms de schéma d'URI, tels que définis par <scheme> au paragraphe 3.1, forment un espace de nom enregistré qui est géré par l'IANA conformément aux procédures définies dans [BCP35]. Aucune action de l'IANA n'est exigée par le présent document.

9. Remerciements

La présente spécification est tirée des RFC 2396 [RFC2396], RFC 1808 [RFC1808], et RFC 1738 [RFC1738] ; Les remerciements figurant dans ces documents s'appliquent toujours. Elle incorpore aussi la mise à jour (avec corrections) des littéraux IPv6 dans la syntaxe d'hôte, telle que définie par Robert M. Hinden, Brian E. Carpenter, et Larry Masinter dans [RFC2732]. De plus, nous remercions chaleureusement de leurs contributions Gisle Aas, Reese Anschultz, Daniel Barclay, Tim Bray, Mike Brown, Rob Cameron, Jeremy Carroll, Dan Connolly, Adam M. Costello, John Cowan, Jason Diamond, Martin Düerst, Stefan Eissing, Clive D.W. Feather, Al Gilman, Tony Hammond, Elliotte Harold, Pat Hayes, Henry Holtzman, Ian B. Jacobs, Michael Kay, John C. Klensin, Graham Klyne, Dan Kohn, Bruce Lilly, Andrew Main, Dave McAlpin, Ira McDonald, Michael Mealling, Ray Merkert, Stephen Pollei, Julian Reschke, Tomas Rokicki, Miles Sabin, Kai Schaetzl, Mark Thomson, Ronald Tschalaer, Norm Walsh, Marc Warne, Stuart Williams, et Henry Zongaro.

10. Références

10.1 Références normatives

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

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[STD63] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[UCS] International Organization for Standardization, "Information Technology - Universal Multiple-Octet Coded Character Set (UCS)", ISO/IEC 10646:2003, December 2003.

10.2. Références informatives

[BCP19] Freed, N. and J. Postel, "IANA Charset Registration Procedures", BCP 19, RFC 2978, October 2000.

[BCP35] Petke, R. and I. King, "Registration Procedures for URL Scheme Names", BCP 35, RFC 2717, November 1999.

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, October 1985.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[RFC1535] Gavron, E., "A Security Problem and Proposed Correction With Widely Deployed DNS Software", RFC 1535, October 1993.

[RFC1630] Berners-Lee, T., "Universal Identifiants de ressource in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web", RFC 1630, June 1994.

[RFC1736] Kunze, J., "Functional Recommendations for Internet Ressource Locators", RFC 1736, February 1995.

[RFC1737] Sollins, K. and L. Masinter, "Functional Requirements for Uniform Ressource Names", RFC 1737, December 1994.

[RFC1738] Berners-Lee, T., Masinter, L., and M. McCahill, "Uniform Ressource Locators (URL)", RFC 1738, December 1994.

[RFC1808] Fielding, R., "Relative Uniform Ressource Locators", RFC 1808, June 1995.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, May 1997.

[RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Identifiants de ressource (URI): Generic Syntax", RFC 2396, August 1998.

[RFC2518] Goland, Y., Whitehead, E., Faizi, A., Carter, S., and D. Jensen, "HTTP Extensions for Distributed Authoring -- WEBDAV", RFC 2518, February 1999.

[RFC2557] Palme, J., Hopmann, A., and N. Shelness, "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", RFC 2557, March 1999.

[RFC2718] Masinter, L., Alvestrand, H., Zigmond, D., and R. Petke, "Guidelines for new URL Schemes", RFC 2718, November 1999.

[RFC2732] Hinden, R., Carpenter, B., and L. Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, December 1999.

[RFC3305] Mealling, M. and R. Denenberg, "Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Identifiants de ressource (URIs), URLs, and Uniform Ressource Names (URNs): Clarifications and Recommendations", RFC 3305, August 2002.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, April 2003.

[Siedzik] Siedzik, R., "Semantic Attacks: What's in a URL?", April 2001, <http://www.giac.org/practical/gsec/Richard_Siedzik_GSEC.pdf>.

Appendice A. ABNF raisonné pour URI

```
URI           = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part     = "//" authority path-abempty
              / path-absolute
              / path-rootless
              / path-empty

URI-reference = URI / relative-ref

absolute-URI  = scheme ":" hier-part [ "?" query ]

relative-ref  = relative-part [ "?" query ] [ "#" fragment ]

relative-part = "//" authority path-abempty
              / path-absolute
              / path-noscheme
              / path-empty

scheme        = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )

authority     = [ userinfo "@" ] host [ ":" port ]
userinfo      = *( unreserved / pct-encoded / sub-delims / ":" )
host          = IP-literal / IPv4address / reg-name
port          = *DIGIT

IP-literal    = "[" ( IPv6address / IPvFuture  ) "]"

IPvFuture     = "v" 1*HEXDIG "." 1*( unreserved / sub-delims / ":" )

IPv6address   =                            6( h16 ":" ) ls32
              /                       "::" 5( h16 ":" ) ls32
              / [               h16 ] "::" 4( h16 ":" ) ls32
              / [ *1( h16 ":" ) h16 ] "::" 3( h16 ":" ) ls32
              / [ *2( h16 ":" ) h16 ] "::" 2( h16 ":" ) ls32
              / [ *3( h16 ":" ) h16 ] "::"    h16 ":"   ls32
              / [ *4( h16 ":" ) h16 ] "::"              ls32
              / [ *5( h16 ":" ) h16 ] "::"              h16
              / [ *6( h16 ":" ) h16 ] "::"

h16           = 1*4HEXDIG
ls32          = ( h16 ":" h16 ) / IPv4address
IPv4address   = dec-octet "." dec-octet "." dec-octet "." dec-octet

dec-octet     = DIGIT                 ; 0-9
              / %x31-39 DIGIT         ; 10-99
              / "1" 2DIGIT            ; 100-199
              / "2" %x30-34 DIGIT     ; 200-249
              / "25" %x30-35          ; 250-255

reg-name      = *( unreserved / pct-encoded / sub-delims )

path          = path-abempty    ; commence par "/" ou est vide
              / path-absolute   ; commence par "/" mais pas "//"
              / path-noscheme   ; commence par un segment qui n'est pas deux points
              / path-rootless   ; commence par un segment
              / path-empty      ; caractères zéro

path-abempty  = *( "/" segment )
path-absolute = "/" [ segment-nz *( "/" segment ) ]
path-noscheme = segment-nz-nc *( "/" segment )
path-rootless = segment-nz *( "/" segment )
```

```
path-empty    = 0<pchar>

segment       = *pchar
segment-nz    = 1*pchar
segment-nz-nc = 1*( unreserved / pct-encoded / sub-delims / "@" )
              ; segment de longueur différente de zéro sans aucun deux points ":"

pchar         = unreserved / pct-encoded / sub-delims / ":" / "@"

query         = *( pchar / "/" / "?" )

fragment      = *( pchar / "/" / "?" )

pct-encoded   = "%" HEXDIG HEXDIG

unreserved    = ALPHA / DIGIT / "-" / "." / "_" / "~"
reserved      = gen-delims / sub-delims
gen-delims    = ":" / "/" / "?" / "#" / "[" / "]" / "@"
sub-delims    = "!" / "$" / "&" / "'" / "(" / ")"
              / "*" / "+" / "," / ";" / "="
```

Appendice B. Analyse d'une référence d'URI avec une expression régulière

Comme l'algorithme "le premier qui correspond l'emporte" est identique à la méthode de résolution des ambiguïtés "gourmande" utilisée par les expressions régulières de POSIX, il est naturel et banal d'utiliser une expression régulière pour analyser les cinq composants potentiels d'une référence d'URI.

La ligne suivante est l'expression régulière pour séparer une référence d'URI bien conformée selon ses composants.

```
^(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?
 12            3  4          5       6  7        8 9
```

Les nombres de la seconde ligne ci-dessus ne sont là que pour faciliter la lecture ; ils indiquent les points de référence pour chaque sous-expression (c'est-à-dire, chaque appariement de parenthèse). On se réfère à la valeur satisfaite pour la sous-expression <n> comme $<n>. Par exemple, satisfaire à l'expression ci-dessus pour http://www.ics.uci.edu/pub/ietf/uri/#Related donne la correspondance de la sous-expression suivante:

```
$1 = http:
$2 = http
$3 = //www.ics.uci.edu
$4 = www.ics.uci.edu
$5 = /pub/ietf/uri/
$6 = <undefined>
$7 = <undefined>
$8 = #Related
$9 = Related
```

où <undefined> indique que le composant n'est pas présent, comme c'est le cas pour le composant query dans l'exemple ci-dessus. On peut donc déterminer la valeur des cinq composants comme

```
scheme    = $2
authority = $4
path      = $5
query     = $7
fragment  = $9
```

Dans le sens opposé, on peut recréer une référence d'URI à partir de ses composants en utilisant l'algorithme du paragraphe 5.3.

Appendice C. Délimitation d’un URI dans son contexte

Les URI sont souvent transmis dans des formats qui ne donnent pas un contexte clair pour leur interprétation. Par exemple, il y a de nombreuses occasions où un URI est inclus en texte clair ; à titre d’exemple on citera le texte envoyé dans un email, les nouvelles USENET, et sur papier imprimé. Dans de tels cas, il est important d’être capable de délimiter l’URI du reste du texte, et en particulier des signes de ponctuation qui pourraient être confondus avec des parties de l’URI.

En pratique, les URI sont délimités de diverses façons, mais généralement par des guillemets "http://example.com/", des signes inférieur/supérieur à <http://example.com/>, ou en utilisant simplement un espace http://example.com/

Ces enveloppes ne font pas partie de l’URI.

Dans certains cas, des espaces blancs supplémentaires (espaces, saut de ligne, signe de tabulations, etc.) ont pu être ajoutés pour couper à la ligne un long URI. Les espaces blancs devraient être ignorés lors de l’extraction de l’URI.

Aucun espace blanc ne devrait être introduit après un caractère tiret ("-"). Comme certains jeux de clavier ou imprimantes peuvent (par erreur) introduire un tiret à la fin d’une ligne lors du passage à la ligne, l’interprète d’un URI contenant un saut de ligne immédiatement après un tiret devrait ignorer tous les espaces blancs autour du saut de ligne et être attentif au fait que le tiret peut en réalité être ou n’être pas une partie de l’URI.

L’utilisation des signes inférieur/supérieur à <> autour de chaque URI est spécialement recommandé comme style de délimitation pour une référence contenant des espaces blancs incorporés.

Le préfixe "URL:" (avec ou sans espace à la suite) était autrefois recommandé comme moyen d’aider à distinguer un URI d’autres désignations entre guillemets, mais il n’est plus d’usage courant en pratique et n’est donc plus recommandé.

Pour des besoins de robustesse, un logiciel qui accepte un URI tapé par l’utilisateur devrait essayer de reconnaître et dépouiller les espaces blancs aussi bien délimiteurs qu’incorporés.

Par exemple, le texte

Oui, Jim, je l’ai trouvé sous "http://www.w3.org/Addressing/", mais tu peux probablement l’avoir à <ftp://foo.example.com/rfc/>.

Noter que l’avertissement dans http://www.ics.uci.edu/pub/ietf/uri/historical.html#WARNING. contient les références d’URI

http://www.w3.org/Addressing/
ftp://foo.example.com/rfc/
http://www.ics.uci.edu/pub/ietf/uri/historical.html#WARNING

Appendice D. Changements par rapport à RFC 2396

D.1. Ajouts

On a introduit une règle ABNF pour URI afin de correspondre à une utilisation commune du terme : un URI absolu avec fragment facultatif.

Les littéraux IPv6 (et ultérieurs) ont été ajoutés à la liste des identifiants possibles pour la portion hôte d'un composant autorité, comme décrit par [RFC2732], avec l'ajout de "[" et "]" à l'ensemble réservé et un fanion de version pour anticiper des versions futures de littéraux IP. Les crochets sont maintenant spécifiés comme réservés à l'intérieur du composant autorité et ne sont plus admis en dehors de leur utilisation comme délimiteurs pour un littéral IP au sein de l'hôte. Dans le but d'effectuer ces changements sans modifier la définition technique des composants path, query, et fragment, ces règles ont été redéfinies pour spécifier directement les caractères admis.

Comme la [RFC2732] renvoie à la [RFC3513] pour la définition d'une adresse littérale IPv6, qui malheureusement, n'a pas de description ABNF de l'adresse IPv6, on a créé une nouvelle règle ABNF pour IPv6address qui satisfait aux représentations textuelles définies au paragraphe 2.2 de [RFC3513]. De même, la définition de IPv4address a été améliorée afin de limiter chaque octet décimal à la gamme 0-255.

La Section 6, sur la normalisation et la comparaison d'URI, a été complètement réécrite et étendue en utilisant une contribution de Tim Bray et une discussion au sein du W3C Groupe d'architecture technique.

D.2. Modifications

La syntaxe BNF ad hoc de la RFC 2396 a été remplacée par l'ABNF de [RFC2234]. Cette modification a obligé à renommer tous les noms de règles qui incluaient autrefois des caractères de souligné en les remplaçant par un tiret. De plus, un certain nombre de règles de syntaxe ont été éliminées ou simplifiées pour rendre l'ensemble de la grammaire plus compréhensible. Les spécifications qui se réfèrent à des règles grammaticales obsolètes peuvent être comprises en remplaçant ces règles selon le tableau ci-après :

Règle obsolète	Traduction
absoluteURI	absolute-URI
relativeURI	relative-part ["?" query]
hier_part	("//" authority path-abempty /
	path-absolute) ["?" query]
opaque_part	path-rootless ["?" query]
net_path	"//" authority path-abempty
abs_path	path-absolute
rel_path	path-rootless
rel_segment	segment-nz-nc
reg_name	reg-name
server	authority
hostport	host [":" port]
hostname	reg-name
path_segments	path-abempty
param	*<pchar excluding ";">
uric	unreserved / pct-encoded / ";" / "?" / ":"
	/ "@" / "&" / "=" / "+" / "$" / "," / "/"
uric_no_slash	unreserved / pct-encoded / ";" / "?" / ":"
	/ "@" / "&" / "=" / "+" / "$" / ","
mark	"-" / "_" / "." / "!" / "~" / "*" / "'"
	/ "(" / ")"
escaped	pct-encoded
hex	HEXDIG
alphanum	ALPHA / DIGIT

L’utilisation des règles obsolètes ci-dessus pour la définition de la syntaxe spécifique de schéma est déconseillée.

La Section 2, sur les caractères, a été réécrite pour expliquer quels caractères sont réservés, quand ils sont réservés, et pourquoi ils sont réservés, et même quand ils ne sont pas utilisés comme délimiteurs par la syntaxe générique. Les caractères de ponctuation qui sont typiquement non sûrs au décodage, y compris le point d’exclamation ("!"), l’astérisque ("*"), les guillemets ("'"), et les ouverture et fermeture de parenthèses ("(" et ")"), ont été transplantés dans l’ensemble réservé afin d’éclaircir la distinction entre réservé et non réservé et, on l’espère, la question la plus courante des concepteurs de schémas. De même, la section sur les caractères codés en pourcentage a été réécrite, et les normalisateurs d’URI ont maintenant la faculté de décoder tout octet codé en pourcentage qui correspond à des caractères non réservés. En général, les termes "escaped" et "unescaped" ont été remplacé par "percent-encoded" et "decoded", respectivement, pour réduire la confusion avec d’autres formes de mécanismes d’échappement (escape).

L’ABNF pour URI et référence d’URI a été redessiné pour le rendre plus accessible pour les analyseurs LALR et en réduire la complexité. Il en résulte que la description formelle de la syntaxe a été supprimée, ainsi que les uric, uric_no_slash, opaque_part, net_path, abs_path, rel_path, path_segments, rel_segment, et les règles de ponctuation. Toutes les références à des URI "opaques" ont été remplacées par une meilleure description de la façon dont le composant path peut être opaque pour la hiérarchie. La règle relativeURI a été remplacée par relative-ref pour éviter une confusion possible pour savoir si c’est un sous-ensemble d’URI. L’ambiguïté concernant l’analyse de URI-reference comme un URI ou une relative-ref avec deux points dans le premier segment a été éliminée grâce à l’utilisation de cinq règles de correspondance de chemin séparées.

L’identifiant de fragment a été remis dans la section sur les composants de la syntaxe générique et dans les règles d’URI et de relative-ref, bien qu’il reste exclu de absolute-URI. Le caractère dièse ("#") a été remis dans l’ensemble réservé par suite de sa réintégration dans la syntaxe de fragment.

L’ABNF a été corrigé pour permettre que le composant path soit vide. Cela permet aussi à un absolute-URI de n’avoir rien après le "scheme:", comme c’est maintenant la pratique avec l’espace de nom "dav:" [RFC2518] et avec le schéma "about:" utilisé en interne par de nombreuses mises en œuvre de navigateur WWW. L’ambiguïté sur les frontières entre authority et path a été éliminée par l’utilisation de cinq règles de correspondance de chemin séparées.

Les autorités de nommage fondé sur l’enregistrement qui utilisent la syntaxe générique sont maintenant définies au sein de la règle host. Ce changement permet aux mises en œuvre courantes, où quel que soit le nom fourni, il alimente tout simplement le mécanisme de résolution de nom local, pour être cohérent avec la spécification. Cela supprime le besoin de re-spécifier ici les formats de noms DNS. De plus, cela permet au composant host de contenir des octets codés en pourcentage, ce qui est nécessaire pour permettre la fourniture aux URI de noms de domaine internationalisés, traités dans leur codage de caractère d’origine dans les couches d’application situées au-dessus du traitement d’URI, et passés à une bibliothèque IDNA en tant que nom enregistré dans le codage de caractère UTF-8. Les règles server, hostport, hostname, domainlabel, toplabel, et alphanum ont été supprimées.

L’algorithme de résolution des références croisées de la [RFC2396] a été réécrit en pseudocode pour cette révision afin d’améliorer la clarté et résoudre les questions suivantes :

- La [RFC2396] paragraphe 5.2, étape 6a, ne réussissait pas à prendre en compte un URI de base sans path.

- Restaurer le comportement de [RFC1808] où, si la référence contient un path vide et un composant query défini, l’URI cible hérite du composant path de l’URI de base.

- La détermination de savoir si une référence d’URI est une référence du même document a été séparée de l’analyseur d’URI, simplifiant l’interface de traitement de l’URI au sein des applications d’une façon cohérente avec l’architecture interne des mises en œuvre de traitement d’URI existantes. Cette détermination est maintenant fondée sur la comparaison avec l’URI de base après la transformation d’une référence en forme absolue, plutôt que dans le format de la référence elle même. Cette modification peut aboutir à ce que plus de références soient considérées comme "même-document" dans la présente spécification qu’il n’y en aurait eu selon les règles données dans la RFC 2396, spécialement lorsque la normalisation est utilisée pour réduire les alias. Cependant, cela ne change pas le statut des références de même-document existantes.

O Diviser la routine de fusion de path en deux routines : merge (*fusionner*), pour décrire la combinaison du chemin de l'URI de base avec une référence de relative-path, et remove_dot_segments, pour décrire comment retirer les segments spéciaux "." et ".." d'un chemin composé. L'algorithme remove_dot_segments s'applique maintenant à tous les chemins de référence d'URI afin de correspondre aux mises en œuvre courantes et améliorer la normalisation des URI en pratique. Cette modification n'a d'impact que sur l'analyse des références anormales et des références de même-schéma lorsque l'URI de base a un chemin non hiérarchique.

Index

Adresses des auteurs

Tim Berners-Lee
World Wide Web Consortium
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139
USA

Phone: +1-617-253-5702
Fax: +1-617-258-5999
EMail: timbl@w3.org
URI: http://www.w3.org/People/Berners-Lee/

Roy T. Fielding
Day Software
5251 California Ave., Suite 110
Irvine, CA 92617
USA

Phone: +1-949-679-2960
Fax: +1-949-679-2972
EMail: fielding@gbiv.com
URI: http://roy.gbiv.com/

Larry Masinter
Adobe Systems Incorporated
345 Park Ave
San Jose, CA 95110
USA

Phone: +1-408-536-3024

EMail: LMM@acm.org
URI: http://larry.masinter.net/

Déclaration de droits d'auteurs

Propriété intellectuelle

Remerciements

Le financement de la fonction d'édition des RFC est actuellement fourni par Internet Society.