Groupe de travail Réseau
Request for Comments : 4825
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

J. Rosenberg, Cisco

mai 2007

Protocole d'accès à la configuration du langage de balisage extensible (XCAP)

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
La présente spécification définit le protocole d'accès à la configuration du langage de balisage extensible (XCAP, *Extensible Markup Language (XML) Configuration Access Protocol*). XCAP permet à un client de lire, écrire, et modifier les données de configuration d'application mémorisées en format XML sur un serveur. XCAP transpose les sous arborescences et attributs de document XML en URI HTTP, afin que ces composants puissent être directement accédés par HTTP.

Table des matières

1. Introduction

Dans beaucoup d'applications de communications, comme la voix sur IP, la messagerie instantanée, et présence, il est nécessaire que les serveurs réseau accèdent aux informations par utilisateur dans le processus de service d'une demande. Ces informations par utilisateur résident dans le réseau, mais sont gérées par l'utilisateur d'extrémité lui-même. Cette gestion peut être faite à travers divers points d'accès, incluant la Toile, un combiné sans fil, ou une application de PC.

Il y a beaucoup d'exemples d'informations par utilisateur. L'une d'elles est la politique d'autorisation de présence [RFC3856] qui définit les règles par lesquelles les observateurs *(watchers)* sont autorisés à souscrire à une présentité, et à quelles informations ils sont admis à accéder. Un autre est les listes de présence, qui sont des listes d'utilisateurs dont la présence est désirée par un observateur [RFC2778]. Une façon d'obtenir les informations de présence pour la liste est de souscrire à une ressource qui représente cette liste [RFC4662]. Dans ce cas, le serveur de liste de ressources (RLS, *Resource List Server*) exige l'accès à cette liste afin de traiter une demande SUBSCRIBE SIP [RFC3261], [RFC3265] pour elle. Une autre façon d'obtenir la présence pour les utilisateurs sur la liste est qu'un observateur souscrive à chaque utilisateur individuellement. Dans ce cas, il est pratique d'avoir un serveur qui mémorise la liste, et quand le client s'amorce, il va chercher la liste chez le serveur. Cela permet à un utilisateur d'accéder aux listes de ressources de différents clients.

La présente spécification décrit un protocole qui peut être utilisé pour manipuler ces données par utilisateur. Il est appelé le protocole d'accès à la configuration du langage de balisage extensible (XCAP). XCAP est un ensemble de conventions pour transposer les documents XML et les composants de document en URI HTTP, de règles sur comment la modification d'une ressource en affecte une autre, de contraintes de validation de données, et des politiques d'autorisation associées à l'accès à ces ressources. À cause de cette structure, les primitives HTTP normales peuvent être utilisées pour manipuler les données. XCAP est fortement fondé sur des idées empruntées au protocole d'accès à la configuration d'application (ACAP,

Application Configuration Access Protocol) [RFC2244], mais il n'en est pas une extension, et n'a aucune dépendance à son égard. Comme ACAP, XCAP est destiné à prendre en charge les besoins de configuration pour diverses applications, plutôt qu'une en particulier.

XCAP n'a pas ete conçu comme un protocole de recherche XML d'utilisation générale, un protocole de mise à jour de base de données XML, ni un protocole de configuration d'utilisation générale, fondé sur XML pour les éléments de réseau.

2. Vue d'ensemble du fonctionnement

Chaque application (une application se réfère à un cas d'utilisation qui implique une collection de données et leur sémantique associée) qui utilise XCAP spécifie un usage d'application (Section 5). Cet usage d'application définit le schéma XML [XML-Struct] pour les données utilisées par l'application, ainsi que les autres pièces clés d'information. La tâche principale de XCAP est de permettre aux clients de lire, écrire, modifier, créer, et supprimer les éléments de ces données. Ces opérations sont prises en charge en utilisant HTTP/1.1 [RFC2616]. Un serveur XCAP agit comme dépositaire de collections de documents XML. Il va y avoir des documents mémorisés pour chaque application. Au sein de chaque application, il y a des documents mémorisés pour chaque utilisateur. Chaque utilisateur peut avoir de multiples documents pour une application particulière. Pour accéder à un composant d'un de ces documents, XCAP définit un algorithme pour construire un URI qui peut être utilisé pour référencer ce composant. Les composants se réfèrent à tout élément ou attribut au sein du document. Donc, les URI HTTP utilisés par XCAP pointent sur un document, ou sur des éléments d'informations qui sont d'une granularité plus fine que celle du document XML lui-même. Une ressource HTTP qui suit les conventions de désignation et les contraintes de validation définies ici est appelée une ressource XCAP.

Comme les ressources XCAP sont aussi des ressources HTTP, on peut y accéder en utilisant les méthodes HTTP. La lecture d'une ressource XCAP est accomplie avec la commande GET HTTP ; en créer ou modifier une est fait avec la commande PUT HTTP, et supprimer une des ressources est fait avec un DELETE HTTP. Les ressources XCAP ne représentent pas de descriptifs de traitement ; par suite, les opérations POST sur les URI HTTP qui représentent des ressources XCAP ne sont pas définies. Les propriétés que HTTP associe aux ressources, comme les étiquettes d'entité, s'appliquent aussi aux ressources XCAP. Bien sûr, les étiquettes d'entité sont particulièrement utiles dans XCAP, car elles permettent d'effectuer un certain nombre d'opérations conditionnelles.

Les documents XML qui sont équivalents pour les besoins de beaucoup d'applications peuvent différer dans leur représentation physique. Avec les ressources XCAP, la forme canonique avec commentaires [XML-Canon] d'un document XML détermine l'équivalence logique. En d'autres termes, la spécification canonique détermine comment les espaces significatives DOIVENT être traitées. Elle implique aussi que, par exemple, de nouveaux attributs insérés peuvent apparaître dans n'importe quel ordre au sein de la représentation physique.

3. Terminologie

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans la [RFC2119] et indiquent les niveaux d’exigence pour les mises en œuvre conformes.

4. Définitions

Les termes suivants sont utilisés tout au long du présent document :

Application : collection de composants logiciels au sein d'un réseau dont le fonctionnement dépend des données gérées et mémorisées sur un serveur XCAP.

Arborescence globale : URI qui représente le parent de tous les documents globaux pour un usage d'application particulier au sein d'une racine XCAP particulière.

Client XCAP : client HTTP qui comprend comment suivre les contraintes de désignation et de validation définies dans la présente spécification.

Conventions de désignation : partie d'nn usage d'application qui spécifie les URI bien connus utilisés par une application, ou plus généralement, spécifie les URI auxquels accède normalement une application durant son traitement.

Identifiant d'utilisateur XCAP (XUI, *XCAP User Identifier*) : c'est une chaîne, valide comme élément de chemin dans un URI HTTP, qui est associée à chaque utilisateur desservi par le serveur XCAP.

Identifiant unique d'application (AUID) : identifiant univoque dans l'espace de noms des identifiants uniques d'application créé par la présente spécification, qui différencie les ressources XCAP auxquelles accède une application des ressources XCAP auxquelles accède une autre application.

Insertion positionnelle : opération PUT qui résulte en l'insertion d'un nouvel élément dans un document de telle sorte que sa position, par rapport aux autres enfants du même parent, soit établie par le client.

Racine XCAP : contexte qui contient tous les documents à travers tous les usages d'application et les utilisateurs qui sont gérés par le serveur.

Répertoire d'accueil : URI qui représente le parent pour tous les documents pour un utilisateur particulier pour un usage d'application particulier au sein d'une racine XCAP particulière.

Ressource XCAP : ressource HTTP représentant un document XML, un élément au sein d'un document XML, ou un attribut d'un élément au sein d'un document XML qui suit les contraintes de désignation et de validation de XCAP.

Serveur XCAP : serveur HTTP qui comprend comment suivre les contraintes de désignation et de validation définies dans la présente spécification.

Sélecteur de document : séquence de segments de chemin, chaque segment étant séparé par un caractère "/", qui identifie le document XML choisi au sein d'une racine XCAP.

Sélecteur de nœud : séquence de segments de chemin, chaque segment étant séparé par un caractère "/", qui identifie le nœud XML (élément ou attribut) choisi au sein d'un document.

Séparateur de sélecteur de nœud : un seul segment de chemin égal aux deux caractères tilde "~~" qui est utilisé pour séparer le sélecteur de document du sélecteur de nœud au sein d'un URI HTTP.

URI de document : URI HTTP qui contient la racine XCAP et le sélecteur de document, résultant en le choix d'un document spécifique. Par suite, effectuer une commande GET sur le document d'URI va restituer le document.

URI de nœud : URI HTTP qui contient la racine XCAP, le sélecteur de document, le séparateur de sélecteur de nœud, et le sélecteur de nœud, résultant en la sélection d'un nœud XML spécifique.

URI de racine XCAP : URI HTTP qui représente la racine XCAP. Bien que ce soit un URI syntaxiquement valide, l'URI de racine XCAP ne correspond pas à une ressource réelle sur un serveur XCAP. Les ressources réelles sont créées en ajoutant des informations de chemin supplémentaires à l'URI de racine XCAP.

Usage d'application : informations détaillées sur l'interaction d'une application avec le serveur XCAP.

5. Usages d'application

Chaque ressource XCAP sur un serveur est associée à une application. Afin qu'une application utilise ces ressources, des conventions spécifique d'application doivent être spécifiées. Ces conventions incluent le schéma XML qui définit la structure et les contraintes des données, les URI bien connus pour amorcer l'accès aux données, et ainsi de suite. Toutes ces conventions spécifique d'application sont définies par l'usage d'application.

5.1 Identifiant unique d'application (AUID)

Chaque usage d'application est associé à un nom, appelé un identifiant unique d'application (AUID, *Application Unique ID*). Ce nom identifie de façon univoque l'usage d'application au sein de l'espace de noms des usages d'application, et est différent des AUID utilisés par les autres applications. Les AUID existent dans un des deux espaces de noms. Le premier

espace de noms est l'espace de noms de l'IETF. Cet espace de noms contient un ensemble de jetons, dont chacun est enregistré par l'IANA. Ces enregistrements se font avec la publication de RFC sur la voie de la normalisation [RFC2434], sur la base des lignes directrices de la Section 15. Le second espace de noms est l'espace de noms de fabricant-propriétaire. Chaque AUID dans cet espace de noms est préfixé avec l'inverse du nom de domaine de l'organisation qui a créé l'AUID, suivi d'un point, suivi par un jeton défini par le fabricant. Par exemple, le domaine exemple.com peut créer un AUID avec la valeur "com.exemple.foo" mais ne peut pas en créer un avec la valeur "org.exemple.foo". Les AUID dans l'espace de noms du fabricant n'ont pas besoin d'être enregistrés par l'IANA. L'espace de noms de fabricant est aussi destiné à être utilisé dans des environnements de laboratoire où aucun registre central n'est nécessaire. La syntaxe desAUID, exprimée en ABNF [RFC4234] (et en utilisant du BNF défini dans la [RFC3986]), est :

```
AUID =  global-a-uid / vendor-a-uid
global-a-uid =  a-uid
a-uid = 1*a-uid-char
vendor-a-uid = rev-hostname "." a-uid
rev-hostname = toplabel *( "." domainlabel  )
domainlabel = alphanum / alphanum *( alphanum / "-" ) alphanum
toplabel = ALPHA / ALPHA *( alphanum / "-" ) alphanum
a-uid-char = a-uid-unreserved / pct-encoded / sub-delims / ":" / "@"
                        ; pct-encoded d'après la RFC 3986
                        ; sub-delims d'après la RFC 3986
alphanum = ALPHA / DIGIT
                        ; DIGIT d'après la RFC 4234
                        ; ALPHA d'après la RFC 4234
a-uid-unreserved = ALPHA / DIGIT / "-" / "_" / "~"
```

Les caractères permis pour la production auid sont un sous ensemble de la production pchar définie dans la RFC 3986. En particulier, elle omet le ".", qui permet à l'auid d'être séparé du nom d'hôte inversé.

5.2 Espace de noms par défaut de document

Afin que le serveur XCAP confronte un URI à un élément ou attribut d'un document, tous les préfixes d'espace de noms XML utilisés dans l'URI doivent être expansés [XML-Noms]. Cette expansion exige un contexte de lien d'espace de noms. Ce contexte transpose les préfixes d'espace de noms en URI d'espace de noms. Elle définit aussi un espace de noms par défaut qui s'applique aux éléments dans l'URI sans préfixe d'espace de noms. Le contexte de lien d'espace de noms vient de deux sources. D'abord, la transposition des préfixes d'espace de noms en URI d'espace de noms est obtenue de l'URI lui-même (voir le paragraphe 6.4). Cependant, l'espace de nom de document par défaut est défini par l'usage d'application lui-même, et s'applique à tous les URI qui référencent des ressources dans cet usage d'application. Tous les usages d'application DOIVENT définir un URI d'espace de noms qui représente l'espace de nom de document par défaut à utiliser dans l'évaluation des URI. L'espace de nom de document par défaut ne s'applique pas aux éléments ou attributs au sein des documents eux-mêmes -- il s'applique seulement à l'évaluation des URI au sein de cet usage d'application. Bien sûr, le terme "espace de nom de document par défaut" est distinct du terme "espace de noms par défaut". Ce dernier a la signification standard dans les documents XML, et le premier se réfère au défaut utilisé dans l'évaluation des URI XCAP. XCAP ne change en aucune façon les mécanismes de détermination de l'espace de noms par défaut au sein des documents XML. Cependant, si un document contient un URI représentant une ressource XCAP, l'espace de nom de document par défaut défini par l'usage d'application s'applique aussi à cet URI.

5.3 Validation de données

Une des responsabilités d'un serveur XCAP est de valider le contenu de chaque ressource XCAP quand un client XCAP essaye d'un modifier un. Ceci est fait en utilisant deux mécanismes. D'abord, tous les usages d'application DOIVENT décrire le contenu de leur document en utilisant le schéma XML [XML-Struct]. L'usage d'application DOIT aussi identifier le type MIME pour les documents conformes à ce schéma.

Malheureusement, les schémas XML ne peuvent pas représenter chaque forme de contrainte de données. Par exemple, un élément XML peut contenir un entier qui définit le nombre maximum d'instances d'un autre élément. Cette contrainte ne peut pas être représentée avec un schéma XML. Cependant, de telles contraintes peuvent être importantes pour l'usage d'application. L'usage d'application définit toutes les contraintes supplémentaires au delà de celles du schéma.

Les contraintes d'unicité sont d'une importance particulière. Dans beaucoup de cas, une application va exiger qu'il y ait

seulement une instance d'un certain élément ou attribut dans une portée particulière. Chaque contrainte d'unicité doit être spécifiée en identifiant le champ, ou les combinaisons de champs, qui doivent être uniques, et ensuite en identifiant la portée dans laquelle l'unicité s'applique. Une portée typique est l'ensemble de tous les éléments d'un certain nom au sein du même parent. Une autre portée typique est l'ensemble de tous les URI valides au sein d'un domaine particulier. Dans certains cas, ces contraintes peuvent être spécifiées en utilisant un schéma XML, qui fournit l'élément <unique> à cette fin. D'autres contraintes d'unicité, comme l'unicité d'URI à travers un domaine, ne peuvent pas être exprimées par un schéma. Que le schéma soit utilisé ou non pour exprimer des exigences d'unicité, l'usage d'application DOIT spécifier toutes les exigences d'unicité quand il définit ses besoins de validation de données.

Par exemple, l'usage d'application de listes de ressources [RFC4826] exige que chaque élément <list> ait une valeur unique pout l'attribut "name" au sein d'un seul parent. Autre exemple, l'usage d'application de services RLS [RFC4826] exige que la valeur de l'attribut "uri" de l'élément <service> soit un URI unique dans le domaine de l'URI.

Les containtes d'URI représentent une autre forme de contrainte. Ce sont des contraintes sur le schéma ou la structure de la partie spécifique du schéma de l'URI. Ces sortes de contraintes ne peuvvent pas être exprimées dans un schéma XML. Si ces contraintes sont importantes pour un usage d'application, elles doivent être invoquées explicitement.

Une autre importante contrainte des données est l'intégrité référentielle. L'intégrité référentielle est importante quand le nom ou la valeur d'un élément ou attribut est utilisé comme clé pour choisir un autre élément ou attribut. Un usage d'application PEUT spécifier des contraintes d'intégrité référentielle . Cependant, les serveurs XCAP ne remplacent pas les systèmes de gestion de base de données relationnelles (RDBMS, *Relational Database Management Systems*) et donc les clients NE DOIVENT PAS dépendre des serveurs pour maintenir l'intégrité référentielle. Les clients XCAP sont chargés de faire tous les changements appropriés aux documents afin de maintenir l'intégrité référentielle.

Une autre contrainte est le codage de caractères. XML permet que les documents soient codés en utilisant plusieurs jeux de caractères différents. Cependant, la présente spécification exige que tous les documents utilisés avec XCAP DOIVENT être codés en utilisant UTF-8. Ceci ne peut pas être changé par un usage d'application.

Les informations de validation des données sont consommées par les deux clients, qui les utilisent pour s'assurer qu'elles construisent des demandes qui vont être acceptées par le serveur, et par les serveurs qui valident les contraintes quand ils reçoivent une demande (à l'exception des contraites d'intégrité de référentiel, qui ne sont pas validées par le serveur).

5.4 Sémantique des données

Pour chaque usage d'application, les données présentes dans le document XML ont une sémantique bien définie. L'usage d'application définit cette sémantique, de sorte qu'un client peut construire de façon appropriée un document afin de réaliser le résultat désiré. Elle n'est pas utilisée par le serveur, car il n'est délibérément pas au courant de la sémantique des données qu'il gère. La sémantique des données est exprimée en langue anglaise par l'usage d'application.

Une sémantique particulièrement importante est l'URI de base qui est à utiliser pour la résolution de toutes les références relatives d'URI pointées sur des ressources XCAP. Comme on l'expose plus loin, les références relatives d'URI qui pointent sur des ressources XCAP ne peuvent pas être résolues en utilisant l'URI restitué comme URI de base. Donc, il appartient à l'usage d'application de spécifier l'URI de base.

5.5 Conventions de dénomination

En plus de définir la signification du document dans le contexte d'une application particulière, un usage d'application doit spécifier comment les applications obtiennent les documents dont elles ont besoin. En particulier, il doit définir les URI bien connus qui sont utilisés à des fins d'amorçage, et documenter toutes les autres conventions sur les URI utilisés par une application. Il devrait aussi documenter comment les documents font référence l'un à l'autre. Ces conventions sont appelées des conventions de dénomination.

Pour beaucoup d'usages d'application, les utilisateurs ont seulement besoin d'un seul document. Dans ce cas, il est RECOMMANDÉ que l'usage d'application exige que ce document soit appelé "index" et existe dans le répertoire d'accueil de l'utililsateur.

Par exemple, l'usage d'application de services de RLS permet à un RLS d'obtenir le contenu d'une liste de ressources quand le RLS reçoit une demande SUBSCRIBE pour un URI SIP qui identifie un service de RLS. L'usage d'application spécifie que la liste des définitions de service est présente dans un document spécifique avec un nom spécifique dans l'arborescence

globale. Cela permet au RLS d'effectuer une seule demande XCAP pour aller chercher la définition de service pour le service associé à l'URI SIP dans une demande SUBSCRIBE.

Les conventions de dénomination sont utilisées par les clients XCAP pour construire leurs URI. Le serveur XCAP ne les utilise pas.

5.6 Interdépendances de ressources

Quand un utilisateur modifie une ressource XCAP, le contenu de beaucoup d'autres ressources est affecté. Par exemple, quand un utilisateur supprime un élément XML dans un document, il le fait en produisant une demande DELETE contre l'URI pour la ressource de l'élément. Cependant, supprimer cet élément supprime aussi tous les éléments fils et leurs attributs, dont chacun est aussi une ressource XCAP. À ce titre, la manipulation d'une ressource affecte l'état des autres ressources.

Pour la pus grande partie, ces interdépendances sont pleinement spécifiées par le schéma XML utilisé par l'usage d'application. Cependant, dans certains usages d'application, il est nécessaire que le serveur mette les ressources en relation entre elles, et une telle relation ne peut pas être spécifiée par un schéma. Cela se produit quand des changements dans un document vont affecter un autre document. Normalement, c'est le cas quand un usage d'application définit un document qui agit comme une collection d'informations définies dans d'autres documents.

Par exemple, quand un utilisateur crée un nouveau service de RLS (c'est-à-dire, il crée un nouvel élément <service> dans un document de services RLS) le serveur ajoute cet élément à une liste globale en lecture seule de services tenus par le serveur dans l'arborescence globale. Le RLS accède à cette liste globale en lecture seule quand il traite une demande SUBSCRIBE SIP.

Les interdépendances de ressources sont utilisées par les clients et les serveurs XCAP.

5.7 Politiques d'autorisation

Par défaut, chaque utilisateur est capable d'accéder (lire, modifier, et supprimer) tous les documents en dessous de leur répertoire d'accueil, et tout utilisateur est capable de lire les documents au sein du répertoire global. Cependant, seuls les utilisateurs de confiance, explicitement provisionnés dans le serveur, peuvent modifier les documents globaux.

L'usage d'application peut spécifier une politique d'autorisation différente qui s'applique à tous les documents associés à cet usage d'application. Un usage d'application peut aussi spécifier si un autre usage d'application est utilisé pour définir les politiques d'autorisation. Un usage d'application pour établir des politiques d'autorisation peut aussi être défini à la suite de la définition de l'usage d'application principal. Dans ce cas, l'usage d'application principal a seulement besoin de spécifier qu'un tel usage sera défini à l'avenir.

Si un usage d'application ne souhaite pas changer la politique d'autorisation par défaut, il peut simplemenl déclarer que la politique par défaut est utilisée.

Les politiques d'autorisation définies par l'usage d'application sont utilisées par le serveur XCAP durant son fonctionnement.

5.8 Extensibilité des données

Un serveur XCAP DOIT comprendre un usage d'application afin de traiter une demande HTTP faite sur une ressource pour cet usage d'application particulier. Cependant, il n'est pas exigé que le serveur comprenne tous les contenus d'un document utilisé par un usage d'application. Il est exigé d'un serveur qu'il comprenne le schéma de base défini par l'usage d'application. Cependant, ces schémas peuvent définir des points d'extensibilité où du nouveau contenu peut être ajouté provenant d'autres espaces de noms et des schémas correspondanrs. Parfois, le serveur va comprendre ces espaces de noms et donc avoir accès à leur schéma, parfois non.

Un serveur DOIT permettre des documents qui contiennent des éléments provenant d'espaces de noms ignorés du serveur. Dans ce cas, le serveur ne peut pas valider si un tel contenu est conforme au schéma ; il va seulement vérifier que le XML est bien formé.

Si un client veut vérifier qu'un serveur prend en charge un espace de noms particulier avant d'operer sur une ressource, il

peut interroger le serveur sur ses capacités en utilisant l'usage d'application Capacités XCAP, discuté à la Section 12.

5.9 Documentation des usages d'application

Les usages d'application sont documentés dans des spécifications qui portent les informations décrites ci-dessis. En particulier, une spécification d'usage d'application DOIT fournir les informations suivantes :

- o Identifiant unique d'application (AUID) : si l'usage d'application est destiné à une utilisation générale sur l'Internet, il DOIT enregistrer l'AUID dans l'arborescence de l'IETF en utilisant les procédures de l'IANA définies à la Section 15.
- o Schéma XML
- o Espace de noms de document par défaut
- o Type MIME
- o Contraintes de validation
- o Sémantique des données
- o Conventions de dénomination
- o Interdépendances de ressources
- o Politiques d'autorisation

5.10 Lignes directrices pour la création d'usages d'application

La principale tâche de conception lors de la création d'un nouvel usage d'application est de définir le schéma. Bien que XCAP puisse être utilisé avec tout document XML, une conception de schéma intelligente va améliorer l'efficacité et l'utilité du document quand il est manipulé avec XCAP.

XCAP fournit trois façons fondamentales de choisir des éléments parmi un ensemble d'apparentés : par l'expansion du nom de l'élément, par sa position, ou par la valeur d'un attribut spécifique. La sélection positionnelle permet toujours à un client d'obtenir exactement ce qu'il veut. Cependant, cela exige que le client mette en antémémoire une copie du document afin de construire le prédicat. De plus, si un client effectue un PUT, cela exige que le client reconstruise le traitement du PUT qu'un serveur va suivre afin de mettre à jour la copie de son antémémoire locale. Autrement, le client va être forcé de refaire un GET pour le document après chaque PUT, ce qui est inefficace. À ce titre, c'est une bonne idée de concevoir des schémas tels que des opérations communes puissent être effectuées sans exiger que le client mette en antémémoire une copie du document.

Sans sélection positionnelle, un client peut prendre l'élément à chaque étape par son nom expansé ou la valeur d'un attribut. De nombreux schémas incluent des éléments qui peuvent être répétés au sein d'un parent (souvent, minOccurs égale zéro ou un, et maxOccurs est sans limite). À ce titre, tous les éléments ont le même nom. Cela laisse la valeur d'attribut comme seul moyen de sélectionner un élément. À cause de cela, si un usage d'application s'attend à ce que l'utilisateur manipule des éléments ou attributs qui sont les descendants d'un élément qui peut se répéter, cet élément DEVRAIT inclure, dans son schéma, un attribut qui puisse être convenablement utilisé comme un index unique. De plus, les conventions de dénomination définies par cet usage d'application DEVRAIENT spécifier explicitement cette contrainte d'unicité.

Les URI sont souvent un bon choix pour un tel index unique. Ils ont des propriétés fondamentales d'unicité, et ont aussi généralement une signification sémantique dans l'usage d'application. Cependant, il faut faire attention quand on utilise un URI comme valeur d'attribut. L'égalité d'URI est généralement complexe. Cependant, l'égalité d'attribut est effectuée par le serveur en utilisant les règles de XML, qui sont fondées sur une comparaison de chaîne sensible à la casse. Donc, XCAP va confronter les URI sur la base de l'égalité lexicale, et non sur une égalité fonctionnelle. Dans ce cas, un usage d'application DEVRAIT considérer avec attention ces implications.

XCAP donne au client la capacité d'opérer sur un seul élément, attribut, ou document à la fois. Par suite, il peut être possible que les opérations courantes que le client peut effectuer exigent une séquence de plusieurs demandes. C'est inefficace, et introduit la possibilité de conditions de défaillance quand un autre client modifie le document au milieu d'une séquence. Dans ce cas, le client va être forcé de détecter cc cas en utilisant des étiquettes d'entité (discutées ci-dessous au paragraphe 7.11) et défaire ses changements précédents. Ceci est très difficile.

Il en résulte que les schémas DEVRAIENT être définis de telle façon que les opérations courantes exigent généralement une seule demande à effectuer. Considérons un exemple. Disons qu'un usage d'application définit des permissions pour que les utilisateurs effectuent certaines opérations. Le schéma peut être conçu de deux façons. Le niveau supérieur de l'arborescence peut identifier des utilisateurs, et dans chaque utilisateur, il peut y avoir les permissions associées à l'utilisateur. Dans une autre conception, le niveau supérieur de l'arborescence identifie chaque permission, et dans cette permission, l'ensemble d'utilisateurs qui l'ont.

Si, dans cet usage d'application, il est courant de changer la permission pour un utilisateur d'une valeur à une autre, la première conception de schéma est meilleure pour XCAP ; elle va exiger un seul PUT pour faire ce changement. Dans le dernier cas, soit le document entier doit être remplacé (ce qui est une seule opération) soit deux opérations PUT doivent survenir -- une pour supprimer l'utilisateur de l'ancienne permission, et une pour ajouter l'utilisateur à la nouvelle permission.

Les conventions de dénomination forment une autre partie clé de la conception d'un usage d'application. L'usage d'application devrait être certain que les clients XCAP savent où "commencer" à restituer et modifier les documents qui les intéressent. Généralement, cela va impliquer la spécification d'un document bien connu à un URI bien connu. Ce document peut contenir des références à d'autres documents que le client doit lire ou modifier.

6. Construction d'URI

Pour manipuler une ressource XCAP, les données doivent être représentée par un URI HTTP. XCAP définit une convention de dénomination spécifique pour construire ces URI. L'URI est construit en enchaînant la racine XCAP au sélecteur de document avec le séparateur de sélecteur de nœud sous une forme codée en pourcentage du sélecteur de nœud. Ceci est suivi par un composant faultatif d'interrogation qui définit les liens d'espace de noms utilisés pour l'évaluation de l'URI. La racine XCAP reste le contexte enclosant dans lequel demeurent toutes les ressources XCAP. Le sélecteur de document est un chemin qui identifie un document au sein de la racine XCAP. Le séparateur de sélecteur de nœud est un segment de chemin avec une valeur de double tilde ("~~") et NE DEVRAIT PAS être codé en pourcentage, comme conseillé au paragraphe 2.3 de la [RFC3986]. Les URI qui contiennent %7E%7E devrait être normalisés en ~~ pour la comparaison ; ils sont équivalents. Le séparateur de sélecteur de nœud séparateur est un élément syntaxique qui sépare le sélecteur de document du sélecteur de nœud. Le sélecteur de nœud est une expression qui identifie un composant du document, comme un élément ou attribut. Il est possible qu'un "~~" apparaisse au titre du sélecteur de nœud lui-même ; dans ce cas, le premier "~~" dans l'URI est le séparateur de sélecteur de nœud.

Les paragraphes qui suivent décrivent ces componsats plus en détails.

6.1 Racine XCAP

La racine de la hiérarchie XCAP est appelée la racine XCAP. Elle définit le contexte dans lequel existent toutes les autres ressources. La racine XCAP est représentée par un URI HTTP, appelé l'URI racine XCAP. Cet URI est un URI HTTP valide ; cependant, il ne pointe sur aucune ressource qui existe actuellement sur le serveur. Son objet est d'identifier la racine de l'arborescence au sein du domaine où tous les documents XCAP sont mémorisés.

Elle peut être tout URI HTTP valide, mais NE DOIT PAS contenir de composant d'interrogation (un URI XCAP complet peut avoir un composant d'interrogation, mais il ne fait pas partie de l'URI racine XCAP). Il est RECOMMANDÉ qu'il soit égal à xcap.domaine, où domaine est le domaine du fournisseur. Par exemple, "http://xcap.exemple.com" pourrait être utilisé comme URI racine XCAP au sein du domaine exemple.com. Normalement, l'URI racine XCAP est provisionné dans les appareils clients. Si il n'est pas explicitement provisionné, les clients DEVRAIENT supposer la forme xcap.domaine, où domaine est le domaine de leur fournisseur de service (pour SIP, ce serait la partie domaine de leur adresse d'hébergement (AOR, *Address-of-Record*)). Un serveur ou domaine PEUT prendre en charge plusieurs URI racine XCAP. Dans ce cas, il fonctionne effectivement comme si il desservait des domaines séparés. Il n'y a jamais de transfert d'informations ou d'interactions entre ressources dans des URI racine XCAP différents.

Quand un client génère une demande HTTP à un URI identifiant une ressource XCAP, les procédures de la RFC 2616 pour la construction de l'URI de demande s'appliquent. En particulier, le composant d'autorité de l'URI ne peut pas être présent dans l'URI de demande si la demande est envoyée directement au serveur d'origine.

L'URI racine XCAP peut aussi être un URI HTTP relatif. Il est de la responsabilité de l'usage d'application de spécifier l'URI de base pour un URI HTTP représentant une ressource XCAP chaque fois qu'un tel URI apparaît au sein d'un document défini par cet usage d'application. Généralement parlant, il n'est pas sûr d'utiliser l'URI restitué comme URI de base. C'est parce que tout URI qui pointe sur un ancêtre d'un élément ou attribut particulier peut contenir du contenu qui inclut cet élément ou attribut. Si cet élément ou attribut contenait une reférence d'URI relatif, elle serait résolue par rapport à tout ce qu'il est arrivé d'être utilisé pour restituer le contenu, et cela va souvent ne pas être l'URI de base défini par l'usage d'application.

6.2 Sélecteur de document

Chaque document au sein de la racine XCAP est identifié par son sélecteur de document. Le sélecteur de document est une séquence de segments de chemin, séparés par une barre oblique ("/"). Ces segments de chemin définissent une structure hiérarchique pour les documents organisateurs au sein de toute racine XCAP. Le premier segment de chemin DOIT être l'AUID XCAP. Ainsi, en continuant l'exemple ci-dessus, tous les documents utilisés par les listes d'application de ressources vont être sous "http://xcap.exemple.com/resource-lists".

o Les mises en œuvre qui utilisent des servlets HTTP devraient savoir que XCAP peut exiger d'elles qu'elles obtiennent l'autorisation de l'administrateur du serveur de placer les ressources dans ce sous ensemble spécifique de l'espace de noms d'URI.

On suppose que chaque application va avoir des données établies par les utilissteurs, et/ou qu'elle va avoir des données globales qui s'appliquent à tous les utilisateurs. Par suite, en dessous de chaque AUID, il y a deux sous arborescences. L'une, appelée "users", contient les documents qui sont applicables à des utilisateurs spécifiques, et l'autre, appelée "global", contient les documents applicables à tous les utilisateurs. La sous arborescence en dessous de "global" est appelée l'arborescence globale. Le segment de chemin après l'AUID DOIT être soit "global", soit "users".

Au sein de l'arborescence "users" sont zéro, une ou plusieurs sous arborescences, dont chacune identifie des documents qui s'appliquent à un utilisateur spécifique. Chaque utilisateur connu du serveur est associé à un nom d'utilisateur, appelé l'identifiant d'utilisation XCAP (XUI, *XCAP User Identifier*). Normalement, un point d'extrémité est provisionné avec la valeur du XUI. Pour les systèmes qui prennent en charge les applications SIP, il est RECOMMANDÉ que le XUI soit égal à l'adresse d'enregistrement (AOR, *Address-of-Record*) pour l'utilisateur (c'est-à-dire, sip:joe@exemple.com). Comme les points d'extrémité SIP connaissent généralement leur AOR, ils vont aussi connaitre leur XUI. Par conséquent, si aucun XUI n'est explicitement provisionné, un agent d'utilisateur SIP DEVRAIT supposer qu'il est égal à son AOR. Ce XUI DOIT être utilisé comme le segment de chemin en dessous du segment "users". Comme l'URI SIP permet des caractères qui ne sont pas permis dans les segments de chemin d'URI HTTP (comme les caractères '?' et '/', qui sont permis dans la partie utilisateur de l'URI SIP) tous ces caractères DOIVENT être codés en pourcentage. La sous arborescence en dessous d'un XUI pour un utilisateur particulier est appelée son répertoire d'accueil. "User" dans ce contexte devrait être interprété largement ; un utilisateur pourrait correspondre à un appareil, par exemple.

XCAP ne définit pas lui-même ce que signifie pour des documents de "s'appliquer" à un utilisateur, au delà d'une spécification d'une politique d'autorisation de base, décrite à la Section 8. Chaque usage d'application peut spécifier des politiques d'autorisation supplémentaires qui dépendent des données utilisées par l'application elle-même.

Le reste du sélecteur de document (le chemin qui suit "global" ou le XUI) pointe sur des documents spécifiques pour cet usage d'application. Les sous répertoires sont permis, mais NE sont PAS RECOMMANDÉS. XCAP ne fournit aucun moyen de créer des sous répertoires ou de faire la liste de leur contenu, limitant donc leur utilité. Si des sous répertoires sont utilisés, il NE DOIT PAS y avoir de document dans un répertoire avec le même nom qu'un sous répertoire.

Le segment final de chemin dans le sélecteur de document identifie le document actuel dans la hiérarchie. C'est équivalent à un nom de fichier, sauf que XCAP n'exige pas que ses ressources de document soient mémorisées comme des fichiers dans un système de fichiers. Cependant, le terme de "filename" est utilisé pour décrire le segment final de chemin dans le sélecteur de document. Dans les sytèmes de fichiers traditionnels, le nom de fichier va avoir une extension de noms de fichier, comme ".xml". Il n'y a rien dans la présente spécification qui exige ou empêche de telles extensions d'être utilisées dans le nom de fichier. Dans certains cas, l'usage d'application va spécifier une convention de dénomination pour les documents, et ces conventions de dénomination peuvent ou non spécifier une extension de fichier. Par exemple, dans l'usage d'application de services de RLS [RFC4826], les documents dans le répertoire d'accueil de l'utilisateur avec le nom de fichier "index" vont être utilisés par le serveur pour calculer l'indice global, qui est aussi un document avec le nom de fichier "index". Sauf des lignes directrices spécifiques dans l'usage d'application, si un utilisateur a un seul document pour un usage d'application particulier, il DEVRAIT être appelé "index".

Quand les conventions de dénomination dans un usage d'application ne mettent pas de contrainte sur les conventions de nom de fichier (ou, plus généralement, sur le sélecteur de document) une application va savoir le nom de fichier (ou plus généralement, le sélecteur de document) parce que il est inclus dans une référence dans un document auquel accède le client. Autre exemple, au sein du document index défini par les services de RLS, l'élément <service> a un élément fils appelé <resource-list> dont le contenu est un URI pointant sur une liste de ressources au sein du répertoire d'accueil des utilisateurs.

Par suite, si l'utilisateur crée un nouveau document, et référence ensuite ce document à partir d'un document bien connu (comme le document index ci-dessus) il n'importe pas que l'utilisateur inclut ou non une extension dans le nom de fichier,

pour autant que l'utilisateur soit cohérent et conserve l'intégrité référentielle.

Par exemple, le segment de chemin "/resource-lists/users/sip:joe@exemple.com/index" est un sélecteur de document. Enchaîner l'URI racine XCAP avec le sélecteur de document produit l'URI HTTP "http://xcap.exemple.com/resource-lists/users/sip:joe@exemple.com/index". Dans cet URI, l'AUID est "resource-lists", et le document est dans l'arborescence de l'utilisateur avec le XUI "sip:joe@exemple.com" avec le nom de fichier "index".

6.3 Sélecteur de nœud

Le sélecteur de nœud spécifie les nœuds spécifiques du document XML auxquels accéder. Un nœud se réfère à un élément XML, à un attribut d'un élément, ou à un ensemble de liens d'espace de noms. Le sélecteur de nœud est une expression qui identifie un élément, attribut, ou ensemble de liens d'espace de noms. Sa grammaire est :

```
sélecteur de nœud = sélecteur d'élément ["/" sélecteur de terminal]
sélecteur de terminal = sélecteur d'attribut / sélecteur d'espace de noms / sélecteur d'extension
sélecteur d'élément = étape *( "/" étape)
étape = par-nom / par-pos / par-attr / par-pos-attr / sélecteur d'extension
par-nom = NomouAutre
par-pos = NomouAutre "[" position "]"
position = 1*CHIFFRE
attr-test = "@" att-nom "=" att-valeur
par-attr = NomouAutre "[" attr-test "]"
par-pos-attr = NomouAutre "[" position "]" "[" attr-test "]"
NomouAutre = QName / "*"                    ; QName d'après l'espace de noms XML
att-nom = QName
att-valeur = AttValeur                      ; d'après la spécification XML
sélecteur d'attribut = "@" att-nom
sélecteur d'espace de noms = "espace de noms::*"
sélecteur d'extension = 1*( %x00-2e / %x30-ff )     ; tout sauf "/"
```

La grammaire de QName est définie dans la spécification des espaces de noms [XML-Noms], et la grammaire de AttValeur est définie dans la spécification XML 1.0 [XML].

Le sélecteur d'extension est inclus aux fins d'extensibilité. Il peut être composé de tout caractère sauf la barre oblique, qui est le délimiteur des étapes. Tout caractère dans une extension qui ne peut pas être représenté dans un URI DOIT être codé en pourcentage avant son placement dans un URI.

Noter que les caractères guillemets, crochet angulaire gauche et droit, qui sont significatifs pour XCAP, ne peuvent pas être représentés directement dans l'URI HTTP. Par suite, ils sont codés en pourcentage quand ils sont placés dans l'URI HTTP. En plus de ces caractères, un caractère apostrophe (') peut être utilisé comme délimiteur au sein des expressions XPath. De plus, comme XML permet des caractères non ASCII, les noms des éléments et attributs peuvent n'être pas directement représentables dans un URI. Tous ces caractères DOIVENT être représentés en les convertissant en une séquence d'octets correspondant en leur représentation en UTF-8, et ensuite en codant en pourcentage cette séquence d'octets.

De même, la spécification XML définit la production QName pour la grammaire des noms d'élément et d'attribut, et la production AttValeur pour les valeurs d'attributs. Malheureusement, les caractères permis par ces productions en incluent qui ne sont pas permis pour pchar, qui est la production pour l'ensemble de caractères permis dans les segments de chemin dans l'URI. La production AttValeur permet beaucoup de ces caractères dans l'ensemble US-ASCII, incluant l'espace. Ces caractères DOIVENT être codés en pourcentage quand ils sont placés dans l'URI. De plus, QName et AttValeur permettent beaucoup de caractères Unicode, en dehors de l'US-ASCII. Quand ces caractères doivent être représentés dans l'URI HTTP, ils sont codés en pourcentage. Pour ce faire, les données devraient être codées d'abord comme octets conformément au codage de caractères UTF-8 [RFC3629], et ensuite seulement ces octets qui ne correspondent pas aux caractères de l'ensemble pchar devraient être codés en pourcentage. Par exemple, le caractère A va être représenté par "A", le caractère LATIN CAPITAL LETTER A WITH GRAVE va être représenté par "%C3%80", et le caractère KATAKANA LETTER A va être représenté par "%E3%82%A2".

Par suite, la grammaire ci-dessus représente les expressions traitées par le serveur XCAP en interne après qu'il a décodé l'URI. Le format sur le réseau est dicté par la [RFC3986]. Dans les discussions et exemples qui suivent, quand les sélecteurs de nœud ne font pas partie d'un URI HTTP, ils sont présentés dans leur format interne avant codage. Si un exemple inclut un sélecteur de nœud au sein d'un URI HTTP, il est présenté dans sa forme codée en pourcentage.

Le sélecteur de nœud se fonde sur les concepts de XPath [XPath]. Bien sûr, l'expression du sélecteur de nœud, avant son codage en pourcentage pour sa représentation dans l'URI HTTP, se trouve être une expression XPath valide. Cependant, XPath fournit un ensemble de fonctionnalités bien plus riches que ce qui est nécessaire ici, et sa largeur introduirait beaucoup de complexité inutile dans XCAP.

Pour déterminer l'élément XML, l'attribut, ou les liens d'espace de noms choisis par le sélecteur de nœud, le traitement commence au nœud racine du document XML. La première étape dans le sélecteur d'élément est alors effectuée. Chaque étape choisit un seul élément XML au sein du contexte courant du document. Le contexte du document est le point dans le document XML à partir duquel une étape spécifique est évaluée. Le contexte du document commence au nœud racine du document. Quand une étape détermine qu'un élément est dans ce contexte, cet élément devient le nouveau contexte pour l'évaluation de la prochaine étape. Chaque étape peut choisir un élément par son nom (expansé) par une combinaison de nom et de valeur d'attribut, par nom et position, ou par nom, position et attribut. Dans tous les cas, le nom peut être muni d'un caractère générique, afin que tous les éléments soient choisis.

L'opération de sélection se fait comme suit. Dans le contexte actuel du document, les enfants de ce contexte sont énumérés dans l'ordre du document. Si le contexte est le nœud racine du document, son élément fils est l'élément racine du document. Si le contexte est un élément, ses enfants sont tous les enfants de cet élément (naturellement). Ensuite, ces éléments dont le nom correspond à NomouAutre sont éliminés. Un nom d'élément est une correspondance si NomouAutre est le caractère générique, ou si il n'est pas un caractère générique, le nom d'élément correspond à NomouAutre. La correspondance est discutée ci-dessous. Le résultat est une liste ordonnée d'éléments.

Les éléments de la liste sont ensuite filtrés par les prédicats, qui sont les expressions entre crochets qui suivent NomouAutre. Chaque prédicat élague de plus les éléments de la liste ordonnée courante. Ces prédicats sont évalués dans l'ordre. Si le contenu du prédicat est une position, l'élément énième position est retenu (c'est-à-dire, on traite "position" comme une variable, et on prend l'élément dont la position est égale à cette variable) et tous les autres sont éliminés. Si il y a moins d'éléments dans la liste que la valeur de position, le résultat est une non correspondance.

Si le contenu du prédicat est un nom et valeur d'attribut, tous les éléments qui possèdent un attribut avec ce nom et cette valeur sont retenus, et tous les autres sont éliminés. Noter que, bien que un document puisse avoir des déclarations d'espace de noms dans les éléments, ces éléments ne peuvent pas être retenus en utilisant une déclaration d'espace de noms comme prédicat. C'est-à-dire, une étape comme "el-name[@xmlns='namespace']" ne vont jamais correspondre à un élément, même si il y a un élément dans la liste qui spécifie un espace de noms par défaut de "namespace". En d'autres termes, un nœud d'espace de noms N'EST PAS un attribut. Si les espaces de noms dans la portée pour un élément sont nécessaires, ils peuvent être choisis en utilisant le sélecteur d'espace de noms décrit ci-dessous. Si il n'y a pas d'élément avec des attributs qui aient le nom et la valeur donnés, le résultat est une non correspondance.

Après l'application des prédicats, le résultat va être une non correspondance, un élément, ou plusieurs éléments. Si le résultat est plusieurs éléments, le sélecteur de nœud est invalide. Chaque étape dans un sélecteur de nœud DOIT produire un seul élément pour former le contexte de la prochaine étape. Ceci est plus restrictif que les expressions générales XPath, qui permettent à un contexte de contenir plusieurs nœuds. Si le résultat est une non correspondance, le sélecteur de nœud est invalide. Le sélecteur de nœud n'est valide que si un seul élément a été retenu. Cet élément devient le contexte pour l'évaluation de la prochaine étape dans l'expression de sélecteur de nœud.

La dernière étape de localisation est soit le sélecteur d'élément précédemment décrit, soit un "sélecteur terminal". Si le sélecteur terminal est un sélecteur d'attribut, le serveur vérifie si il y a un attribut avec le même nom expansé dans le contexte d'élément courant. Si il n'y en a pas, le résultat est considéré comme une non correspondance. Autrement, cet attribut est sélectionné. Si le sélecteur terminal est un sélecteur d'espace de noms, le résultat est égal à l'ensemble des liens d'espace de noms de la portée de l'élément, incluant une possible déclaration d'espace de noms par défaut. La présente spécification définit une syntaxe pour représenter les liens d'espace de noms, de sorte qu'ils peuvent être retournés au client dans une réponse HTTP.

Par suite, une fois que le sélecteur de nœud entier est évalué par rapport au document, le résultat va être une non correspondance, invalide, un seul élément, un seul attribut, ou un ensemble de liens d'espace de noms.

La confrontation des noms d'éléments est effectuée comme suit. L'élément à comparer dans l'étape a son nom expansé comme décrit dans les espaces de noms XML [XML-Noms]. Le nom de l'élément dans l'étape est aussi expansé. Cette expansion exige que tout préfixe d'espace de nom soit converti en son URI d'espace de noms. Le faire exige un ensemble de liens des préfixes aux URI d'espace de noms. Cet ensemble de liens est obtenu du composant d'interrogation de l'URI (voir au paragraphe 6.4). Si le préfixe de QName d'un élément est vide, l'URI correspondant est alors l'URI de l'espace de noms de document par défaut défini par l'usage d'application, ou nul si aucun n'est défini. Les comparaisons sont alors

effectuées comme décrit dans les espaces de noms XML [XML-Noms]. Noter que les expansions de préfixe d'espace de noms décrites ici sont différentes de celles spécifiées dans XPath 1.0, mais sont plus proches de celles actuellement définies dans la spécification XPath 2.0 [XML-Lang].

La confrontation des noms d'attribut se fait d'une façon similaire. L'attribut dans le document a son nom expansé comme décrit dans Espaces de noms XML [XML-Noms]. Si le nom d'attribut dans le sélecteur d'attribut a un préfixe d'espace de noms, son nom est expansé en utilisant les liens d'espace de noms obtenus du composant d'interrogation dans l'URI. Un attribut QName sans préfixe n'est dans aucun espace de noms.

Des commentaires, du contenu de texte (incluant des espaces) et des instructions de traitement peuvent être présents dans un document, mais ne peuvent pas être sélectionnés par les expressions définies ici. Bien sûr, si ces informations sont présentes dans un document, et si un utilisateur choisit un élément XML qui englobe ces données, les informations vont être incluses, par exemple, dans un GET résultant. De plus, les espaces sont respectées par XCAP. Si un client met un PUT sur un élément ou document qui contient des espaces, le serveur conserve ces espaces, et va retourner l'élément ou document au client avec exactement les mêmes espaces. De même, quand un élément est inséré, aucune espace supplémentaire n'est ajoutée autour de l'élément inséré, et l'élément est inséré dans un emplacement très spécifique par rapport à toute espace, commentaire, ou instruction de traitement autour de lui. Le paragraphe 8.2.3 décrit où l'insertion se produit.

Par exemple, considérons le document XML suivant :

```
<?xml version="1.0"?>
<watcherinfo xmlns="urn:ietf:params:xml:ns:watcherinfo" version="0" state="full">
  <watcher-list resource="sip:professor@exemple.net" package="presence">
    <watcher status="active"
          id="8ajksjda7s"
          duration-subscribed="509"
          event="approved">sip:userA@exemple.net</watcher>
    <watcher status="pending"
          id="hh8juja87s997-ass7"
          display-name="Mr. Subscriber"
          event="subscribe">sip:userB@exemple.org</watcher>
  </watcher-list>
</watcherinfo>
```

Figure 3 : Exemple de document XML

En supposant que l'espace de noms de document par défaut pour cet usage d'application est "urn:ietf:params:xml:ns:watcherinfo", le sélecteur de nœud watcherinfo/watcher-list/watcher[@id="8ajksjda7s"] sélectionnerait l'élément XML suivant :

```
<watcher status="active"
   id="8ajksjda7s"
   duration-subscribed="509"
   event="approved">sip:userA@exemple.net</watcher>
```

6.4 Liens d'espace de noms pour le sélecteur

Afin d'expanser les préfixes d'espace de noms utilisés dans le sélecteur de nœud, un ensemble de liens provenant de ces préfixes d'espace de noms en URI d'espace de noms doit être utilisé. Ces liens sont contenus dans le composant d'interrogation de l'URI. Si aucun composant d'interrogation n'est présent, cela signifie que seul l'espace de noms de document par défaut (tel qu'identifié par l'usage d'application) est défini. Le composant d'interrogation est formaté comme une expression xpointer valide [XPointer] après un codage d'URI convenable comme défini au paragraphe 4.1 du cadre Xpointer. Cette expression xpointer DEVRAIT seulement contenir des expressions provenant du schéma xmlns() [XMLns]. Un serveur conforme à la présente spécification DOIT ignorer toute expression xpointer qui n'est pas du schéma xmlns(). Les expressions xpointer xmlns() définissent l'ensemble de liens d'espace de noms utilisés pour évaluer l'URI.

Noter que les expressions xpointer étaient à l'origine conçues pour être utilisées au sein d'identifiants de fragment d'URI. Cependant, dans XCAP, ils sont utilisés dans les composants d'interrogation des URI.

L'exemple suivant montre une opération de confrontation plus complexe, incluant cette fois l'usage de liens d'espace de noms. Considérons le document suivant :

```
<?xml version="1.0"?>
<foo xmlns="urn:test:default-namespace">
  <ns1:bar xmlns:ns1="urn:test:namespace1-uri" xmlns="urn:test:namespace1-uri">
    <baz/>
    <ns2:baz xmlns:ns2="urn:test:namespace2-uri"/>
  </ns1:bar>
  <ns3:hi xmlns:ns3="urn:test:namespace3-uri">
    <there/>
  </ns3:hi>
</foo>
```

Supposons que ce document a un URI de document de "http://xcap.exemple.com/test/users/sip:joe@exemple.com/index", où "test" est l'usage d'application. Cet usage d'application définit un espace de noms de document par défaut de "urn:test:default-namespace". L'URI XCAP :

http://xcap.exemple.com/test/users/sip:joe@exemple.com/index/~~/foo/a:bar/b:baz?xmlns(a=urn:test:namespace1-uri)xmlns(b=urn:test:namespace1-uri)

va choisir le premier élément fils <baz> de l'élément <bar> dans le document. L'URI XCAP :

http://xcap.exemple.com/test/users/sip:joe@exemple.com/index/~~/foo/a:bar/b:baz?xmlns(a=urn:test:namespace1-uri)xmlns(b=urn:test:namespace2-uri)

va choisir le second élément fils <baz> de l'élément <bar> dans le document. L'URI XCAP suivant va aussi choisir le second élément fils <baz> de l'élément <bar> dans le document :

http://xcap.exemple.com/test/users/sip:joe@exemple.com/index/~~/d:foo/a:bar/b:baz?xmlns(a=urn:test:namespace1-uri)xmlns(b=urn:test:namespace2-uri)xmlns(d=urn:test:default-namespace)

7. Opérations du client

Un client XCAP est un client HTTP/1.1 conforme. Les tâches spécifiques de manipulation de données sont accomplies en invoquant le bon ensemble de méthodes HTTP avec le bon ensemble d'en-têtes sur le serveur. Cette section décrit cela en détails.

Dans tous les cas où le client modifie un document, en supprimant ou insérant un document, une ressource d'élément ou d'attribut, le client DEVRAIT vérifier que, si l'opération devait réussir, le document résultant satisfera les contraintes de données définies par l'usage d'application, incluant la validation de schéma. Par exemple, si le client effectue une opération PUT sur "http://xcap.exemple.com/rls-services/users/sip:joe@exemple.com/mespotes", rls-services est l'identifiant d'application unique, et les contraintes définies par lui DEVRAIENT être suivies.

Le clientva savoir quel URI utiliser sur la base des conventions de dénomination décrites par l'usage d'application.

Si le document, après modification, ne satisfait pas les contraintes de données, le serveur va le rejeter avec un 409. La réponse 409 peut contenir un corps XML, formaté en accord avec le schéma du paragraphe 11.2, qui donne plus d'informations sur la nature de l'erreur. Le client PEUT utiliser ces informations pour essayer et altérer la demande afin que, cette fois, elle puisse réussir. Le client NE DEVRAIT PAS simplement ressayer la demande sans en changer quelque aspect.

Dans certains cas, l'usage d'application va dicter une contrainte d'unicité que le client ne peut pas garantir de lui-même. Un exemple en est qu'un URI doit être unique dans un domaine. Normalement, le client n'est pas le propriétaire du domaine, et donc il ne peut pas être sûr que l'URI est unique. Dans ce cas, le client peut soit générer un identifiant suffisamment aléatoire, soit il peut prendre un identifiant "bizarre" dans l'espoir qu'il ne soit pas déjà pris. Dans l'un et l'autre cas, si l'identifiant n'est pas unique, le serveur va rejeter la demande avec un 409 et suggérer des solutions de remplacement que le client peut utiliser pour ressayer. Si le serveur ne suggère pas de solution de remplacement, le client DEVRAIT tenter d'utiliser des identifiants aléatoires avec des quantités croissantes d'aléa.

HTTP spécifie aussi que les demandes PUT et DELETE sont idempotentes. Cela signifie que, si le client effectue un PUT sur un document et qu'il réussit, il peut effectuer le même PUT, et le document résultant va avoir le même aspect. De même, quand un client effectue un DELETE, si il réussit, un DELETE suivant sur le même URI va générer une réponse 404 ; la ressource n'existe plus sur le serveur car elle a été supprimée par l'opération DELETE précédente. Pour conserver cette propriété, le client DEVRAIT construire ses URI de telle sorte que, après que modification a eu lieu, l'URI dans la demande va pointer sur la ressource juste insérée pour PUT (c'est-à-dire, le corps de la demande) et ne va pointer sur rien pour DELETE. Si cette propriété est conservée, le GET sur l'URI dans le PUT va retourner le même contenu (c'est-à-dire, GET(PUT(X)) == x). Cette propriéte implique l'idempotence. Bien qu'une demande puisse encore être idempotente si elle ne possède pas cette propriéte, XCAP ne permet pas ces demandes. Si la demande du client n'a pas cette propriété, le serveur va rejeter la demande avec un 409 et indiquer la condition d'erreur qu'il ne peut pas insérer.

Si le résultat du PUT est une réponse 200 ou 201, l'opération a réussi. Les autres codes de réponse à toute demande, comme une redirection, sont traitées conformément à la [RFC2616].

7.1 Création ou remplacement d'un document

Pour créer ou remplacer un document, le client construit un URI qui fait référence à la localisation où le document doit être placé. Cet URI DOIT être un URI de document, et donc contenir la racine XCAP et le sélecteur de document. Le client invoque alors une méthode PUT sur cet URI.

Le type de contenu MIME DOIT être celui défini par l'usage d'application. Par exemple, il serait "application/rls-services+xml" pour un document de services RLS [RFC4826], et non "application/xml".

Si l'URI de demande identifie un document qui existe déjà dans le serveur, l'opération PUT remplace ce document par le contenu de la demande. Si l'URI de demande n'identifie pas un document existant, le document est créé sur le serveur à cet URI spécifique.

7.2 Supprimer un document

Pour supprimer un document, le client construit un URI qui fait référence au document à supprimer. Cet URI DOIT être un URI de document. Le client invoque alors une opération DELETE sur l'URI pour supprimer le document.

7.3 Aller chercher un document

Comme on s'y attend, aller chercher un document est accompli de façon triviale en effectuant une demande HTTP GET avec l'URI de demande réglé à l'URI du document.

7.4 Créer ou remplacer un élément

Pour créer ou remplacer un élément XML au sein d'un document existant, le client construit un URI dont le sélecteur de document pointe sur le document à modifier. Le sélecteur de nœud DOIT être présent dans l'URI, délimité du sélecteur de document par un séparateur de sélecteur de nœud. Le composant d'interrogation DOIT être présent si le sélecteur de nœud utilise des préfixes d'espace de noms, et dans ce cas, les expressions xmlns() dans le composant d'interrogation DOIVENT définir ces préfixes. Pour créer cet élément au sein du document, le sélecteur de nœud est construit de telle façon qu'il soit une non correspondance à l'égard du document courant, mais si l'élément dans le corps de la demande a été ajouté au document comme désiré par le client, le sélecteur de nœud va sélectionner cet élément. Pour remplacer un élément dans le document, le sélecteur de nœud est construit de telle façon qu'il y ait une correspondance avec l'élément dans le document courant à remplacer, ainsi qu'une correspondanc au nouvel élément (présent dans le corps de la demande PUT) qui doit le remplacer.

Souvent, le client va souhaiter insérer un élément dans un document à une certaine position par rapport aux autres enfants du même parent. Ceci est appelé une insertion positionnelle. Cela arrive souvent parce que le schéma contraint l'endroit où l'élément peut apparaître, ou parce que l'ordre des éléments est significatif dans le schéma. Pour faire cela, le client peut utiliser un sélecteur de nœud de la forme suivante :

```
parent/*[position][unique-attribute-value]
```

Ici, "parent" est une expression pour le parent de l'élément à insérer ; "position" est la position parmi les éléments fils existants de ce parent où le nouvel élément est à insérer ;"unique-attribute-value" est un nom et une valeur d'attribut pour l'élément à insérer, qui est différente de l'élément courant dans "position". Le second prédicat est nécessaire pour que l'expression globale soit une non correspondance quand elle est évaluée par rapport aux enfants actuels. Autrement, le PUT remplacerait l'élément existant dans cette position. Noter qu'en plus d'un caractère générique "*" un QName peut aussi être utilisé comme vérificateur de nœud. Cette logique d'insertion est décrite plus en détails au paragraphe 8.2.3.

Considérons l'exemple de document de la Figure 3. Le client aimerait insérer un nouvel élément <watcher> comme second élément en dessous de <watcher-list>. Cependant, il ne peut pas juste faire un PUT à un URI avec le sélecteur de nœud watcherinfo/watcher-list/*[2] ; ce sélecteur de nœud choisirait le second élément fils existant de <watcher-list> et le remplacerait. Donc, le PUT doit être fait à un URI avec watcherinfo/watcher-list/*[2][@id="hhggff"] comme sélecteur de nœud, où "hhggff" est la valeur de l'attribut "id" du nouvel élément à insérer. Ce sélecteur de nœud est une non correspondance par rapport au document courant, et serait une correpondance pour le nouvel élément si il était inséré comme second élément fils de <watcher-list>.

Le "*" indique que tous les élément fils de <watcher-info> sont à considérer quand on calcule la position pour l'insertion. Si, au lieu d'un caractère générique *, un nom d'élément (QName) était présent, l'expression ci-dessus aurait inséré le nouvel élément comme élément de énième position parmi ceux de même nom expansé (voir au paragraphe 8.2.3 la discussion des règles d'insertion).

Une fois que le client a construit l'URI, il invoque la méthode PUT HTTP. Le contenu dans la demande DOIT être un élément XML. Précisément, il contient l'élément, commençant par le crochet d'ouverture pour l'étiquette de début de cet élément, incluant les attributs et le contenu de cet élément (que ce soit du texte ou un autre élément fils) et se terminant par le crochet de fermeture pour l'étiquette de fin de cet élément. Le type MIME dans la demande DOIT être "application/xcap-el+xml", défini au paragraphe 15.2.1. Si le sélecteur de nœud, quand il évalue par rapport au document courant, résulte en une non correspondance, le serveur effectue une opération de création. Si le sélecteur de nœud, quand il évalue par rapport au document courant, est une correspondance pour un élément dans le document courant, le serveur le remplace par le contenu de la demande PUT. Ce remplacement est complet ; c'est-à-dire, le vieil élément (incluant ses attributs, ses déclarations d'espace de noms et son contenu : nœuds texte, élément, commentaire et instruction de traitement) est retiré, et le nouveau, incluant ses attributs, déclarations d'espace de noms et contenu, est mis à sa place.

Pour être certain que les insertions d'élément ont la propriété GET(PUT(x))==x, le client peut vérifier que les prédicats d'attribut dans le segment final de chemin de l'URI correspondent aux attributs de l'élément dans le corps de la demande. Comme exemple d'une demande qui n'aurait pas cette propriété, et donc ne serait pas idempotente, considérons la demande PUT suivante (les URI reviennent à la ligne pour la lisibilité) :

```
PUT
/rls-services/users/sip:bill@exemple.com/index/~~/rls-services/service%5b@uri=%22sip:bons-amis@exemple.com
%22%5d
HTTP/1.1
Content-Type:application/xcap-el+xml
Host: xcap.exemple.com

<service uri="sip:mespotes@exemple.com">
  <resource-list>http://xcap.exemple.com/resource-lists/users/sip:joe@exemple.com/index/~~/resource-lists/list
%5b@name=%22l1%22%5d
</resource-list>
  <packages>
   <package>presence</package>
  </packages>
</service>
```

Cette demande va échouer avec un 409. L'URI de demande contient un segment final de chemin avec un prédicat fondé sur des attributs : @uri="sip:bons-amis@exemple.com". Cependant, cela ne correspondra pas à la valeur de l'attribut "uri" dans l'élément dans le corps (sip:mespotes@exemple.com).

La propriété GET(PUT(x))==x introduit des limitations aux types d'opérations possibles. Il ne va pas être possible de remplacer un élément par un qui a une nouvelle valeur pour un attribut qui est le seul identifiant unique d'élément, si l'URI contenait un sélecteur de nœud qui utilisait la valeur précédente de cet attribut afin de choisir l'élément. C'est exactement le cas d'utilisation dans l'exemple ci-dessus. Pour contourner cette limitation, le choix peut être fait par position au lieu d'une valeur d'attribut, ou le parent de l'élément à remplacer peut être choisi, et ensuite le corps de l'opération PUT va contenir le

parent, l'enfant à remplacer, et tous les autres apparentés.

7.5 Supprimer un élément

Pour supprimer un élément d'un document, le client construit un URI dont le sélecteur de document pointe sur le document contenant l'élément à supprimer. Le sélecteur de nœud DOIT identifier un seul élément. Le sélecteur de nœud DOIT être présent à la suite du séparateur de sélecteur de nœud, et identifier l'élément spécifique à supprimer. De plus, le sélecteur de nœud DOIT ne correspondre à aucun élément après la suppression de l'élément cible. C'est exigé pour conserver la propriété d'idempotence des suppressions HTTP. Le composant d'interrogation DOIT être présent si le sélecteur de nœud utilise des préfixes d'espace de noms, et dans ce cas les expressions xmlns() dans le composant d'interrogation DOIVENT définir ces préfixes.

Si le client souhaite supprimer un élément dans une position spécifique, c'est appelé une suppression positionnelle. Comme dans une insertion positionnelle, le sélecteur de nœud a la forme suivante :

parent/*[position][unique-attribute-value]

Où "parent" est une expression pour le parent de l'élément à supprimer ; "position" est la position de l'élément à supprimer parmi les éléments fils existants de ce parent, et "unique-attribute-value" est un nom et une valeur d'attribut pour l'élément à supprimer, où le nom et la valeur de cet attribut sont différents de ceux de toute la parentèle de cet élément.

Les suppressions positionnelles sans utiliser un nom et valeur unique d'attribut sont possibles, mais seulement dans des cas limités où l'idempotence est garantie. En particulier, si une opération DELETE se réfère à un élément par le nom et la position seules (parent/elname[n]) ceci n'est permis que quand l'élément à supprimer est le dernier élément parmi tous ses apparentés avec ce nom. De même, si une opération DELETE se réfère à un élément par la position seule (parent/*[n]), ceci n'est permis que quand l'élément à supprimer est le dernier parmi tous les éléments de la parentèle, sans considération du nom.

Le client invoque alors la méthode HTTP DELETE. Le serveu va supprimer l'élément du document (incluant ses attributs, déclarations d'espace de noms, et ses nœuds descendants, comme tous ses enfants).

7.6 Aller chercher un élément

Pour aller chercher un élément d'un document, le client construit un URI dont le sélecteur de document pointe sur le document contenant l'élément à aller chercher. Le sélecteur de nœud DOIT être présent à la suite du séparateur de sélecteur de nœud, et doit identifier l'élément à aller chercher. Le composant d'interrogation DOIT être présent si le sélecteur de nœud utilise des préfixes d'espace de noms, et dans ce cas, les expressions xmlns() dans le composant d'interrogation DOIVENT définir ces préfixes.

Le client invoque alors la méthode GET. La réponse 200 OK va contenir cet élément XML. Précisément, elle contient le contenu du document XML, commençant par le crochet d'ouverture pour l'étiquette de début pour cet élément, et se terminant avec le crochet de fermeture pour l'étiquette de fin pour cet élément. Cela va, par suite, inclure tous les attributs, déclarations d'espace de noms et nœuds descendants : éléments, commentaires, texte, et instructions de traitement de cet élément.

7.7 Créer ou remplacer en attribut

Pour créer ou remplacer un attribut dans un élément existant d'un document, le client construit un URI dont le sélecteur de document pointe sur le document à modifier. Le sélecteur de nœud, suivant le séparateur de sélecteur de nœud, DOIT être présent. Le sélecteur de nœud DOIT être construit de telle façon que, si l'attribut a été créé ou remplacé comme désiré, le sélecteur de nœud va choisir cet attribut. Si le sélecteur de nœud, quand il est évalué par rapport au document courant, résulte en une non correspondance, c'est une opération de création. Si il correspond à un attribut existant, c'est une opération de remplacement. Le composant d'interrogation DOIT être présent si le sélecteur de nœud utilise des préfixes d'espace de noms, et dans ce cas, les expressions xmlns() dans le composant d'interrogation DOIVENT définir ces préfixes.

Le client invoque alors la méthode HTTP PUT. Le contenu défini par la demande DOIT être la valeur de l'attribut, conforme à la grammaire pour AttValue comme définie dans XML 1.0 [XML]. Noter que, à la différence de quand AttValue est présent dans l'URI, il n'y a pas de codage en pourcentage du corps. Cette demande DOIT être envoyée avec le type de contenu "application/xcap-att+xml" comme défini au paragraphe 15.2.2. Le serveur va ajouter l'attribut de façon

telle que, si le sélecteur de nœud est évalué sur le document résultant, il va retourner l'attribut présent dans la demande.

Pour être certain que les insertions d'attribut ont la propriété GET(PUT(x))==x, le client peut vérifier que tout prédicat d'attribut dans le segment de chemin qui sélectionne l'élément dans lequel l'attribut est inséré, correspond à un attribut différent de celui qui est inséré par la demande. Comme exemple d'une demande qui n'aurait pas cette propriété, et donc ne serait pas idempotente, on considère la demande PUT suivante (les URI reviennent à la ligne pour la lisibilité) :

```
PUT
/rls-services/users/sip:bill@exemple.com/index/~~/rls-services/service%5b@uri=%22sip:bons-amis@exemple.com
%22%5d/@uri
HTTP/1.1
Content-Type:application/xcap-att+xml
Host: xcap.exemple.com"sip:mauvais-amis@exemple.com"
```

Cette demande va échouer avec un 409.

Comme avec les insertions et remplacements d'éléments, la propriété GET(PUT(x))==x introduit des limitations sur les remplacements d'attributs. Il ne sera pas possible de remplacer la valeur d'attribut d'un attribut, quand cet attribut est le seul identifiant unique d'élément, et que l'URI contient un sélecteur de nœud qui utilise la précédante valeur de l'attribut pour sélectionner l'élément affecté. C'est le cas d'utilisation de l'exemple ci-dessus. À la place, l'élément peut être sélectionné positionnellement, ou son parent entier être remplacé.

7.8 Supprimer un attribut

Pour supprimer un attribut du document, le client construit un URI dont le sélecteur de document pointe sur le document contenant l'attribut à supprimer. Le sélecteur de nœud DOIT être présent à la suite, par exemple du séparateur de sélecteur de nœud, et s'évaluer comme attribut dans le document à supprimer. Le composant d'interrogation DOIT être présent si le sélecteur de nœud utilise des préfixes d'espace de noms, et dans ce cas les expressions xmlns() dans le composant d'interrogation DOIVENT définir ces préfixes.

Le client invoque alors la méthode HTTP DELETE. Le serveur va supprimer l'attribut du document.

7.9 Aller chercher un attribut

Pour aller chercher un attribut d'un document, le client construit un URI dont le sélecteur de document pointe sur le document contenant l'attribut à aller chercher. Le sélecteur de nœud DOIT être présent à la suite du séparateur de sélecteur de nœud, contenant une expression qui identifie l'attribut dont la valeur est à aller chercher. Le composant d'interrogation DOIT être présent si le sélecteur de nœud utilise des préfixes d'espace de noms, et dans ce cas, les expressions xmlns() dans le composant d'interrogation DOIVENT définir ces préfixes.

Le client invoque alors la méthode GET. La réponse 200 OK va contenir un document "application/xcap-att+xml" avec l'attribut spécifié, formaté conformément à la grammaire de AttValue définie dans les spécifications XML 1.0.

7.10 Aller chercher des liens d'espace de noms

Si un client souhaite insérer un élément ou attribut dans un document, et si cet élément ou attribut fait partie d'un espace de noms déclaré ailleurs dans le document, le client va avoir besoin de savoir les liens d'espace de noms afin de construire le contenu XML dans la demande. Si le client a une copie du document en antémémoire, il va connaître les liens. Cependant, si il n'a pas tout le document en antémémoire, il peut être utile d'aller chercher juste les liens qui sont dans la portée pour un élément, afin de construire une demande PUT suivante.

Pour obtenir ces liens, le client construit un URI dont le sélecteur de document pointe sur le document contenant l'élément dont les liens d'espace de noms sont à aller chercher. Le sélecteur de nœud DOIT être présent à la suite du séparateur de sélecteur de nœud, contenant une expression identifiant les liens d'espace de noms désirés. Le composant d'interrogation DOIT être présent si le sélecteur de nœud utilise des préfixes d'espace de noms, et dans ce cas, les expressions xmlns() dans le composant d'interrogation DOIVENT définir ces préfixes.

Le client invoque alors la méthode GET. La réponse 200 OK va contenir un document "application/xcap-ns+xml" avec les définitions d'espace de noms. Le format de ce document est défini à la Section 10.

Un client ne peut pas établir des préfixes d'espace de noms dans la portée d'un élément. À ce titre, un sélecteur de nœud qui identifie des préfixes d'espace de noms NE DOIT PAS apparaître dans une demande PUT ou DELETE.

7.11 Opérations conditionnelles

La spécification HTTP définit plusieurs champs d'en-tête qui peuvent être utilisés par un client pour rendre conditionnel le traitement de la demande . En particulier, les champs If-None-Match et If-Match permettent à un client de les rendre conditionnelles sur la valeur courante de l'étiquette d'entité pour la ressource. Ces opérations conditionnelles sont particulièrement utiles pour les ressources XCAP.

Par exemple, il est prévu que les clients souhaiteront fréquemment mettre en antémémoire la version courante d'un document. Ainsi, quand le client démarre, il va aller chercher le document en cours sur le serveur et le mémoriser.

Quand il fait ainsi, la réponse GET va contenir l'étiquette d'entité pour la ressource de document. Chaque ressource au sein d'un document conservé par le serveur va partager la même valeur d'étiquette d'entité. Par suite, l'étiquette d'entité retournée par le serveur pour la ressource de document est applicable aux ressources d'élément et d'attribut au sein du document.

Si le client souhaite insérer ou modifier un élément ou attribut au sein du document, mais veut être certain que le document n'a pas été modifié depuis la dernière fois que le client a opéré sur lui, il peut inclure un champ d'en-tête If-Match dans la demande, contenant la valeur de l'étiquette d'entité connue du client pour toutes les ressources du document. Si le document a changé, le serveur va rejeter cette demande avec une réponse 412. Dans ce cas, le client va avoir besoin de purger sa version en antémémoire, d'aller chercher le document entier, et de mémoriser la nouvelle étiquette d'entité retournée par le serveur dans le 200 OK à la demande GET. Il peut alors réessayer la demande, en plaçant la nouvelle étiquette d'entité dans le champ If-Match. Si cela réussit, le champ d'en-tête Etag dans la réponse au PUT contient l'étiquette d'entité pour la ressource qui vient juste d'être insérée ou modifiée. Parce que toutes les ressources dans un document partagent la même valeur d'étiquette d'entité, cette valeur d'étiquette d'entité peut être appliquée à la modification d'autres ressources.

Un client peut aussi supprimer conditionnellement des éléments ou attributs en incluant un champ d'en-tête If-Match dans les demandes DELETE. Noter que les réponses 200 OK à un DELETE vont contenir un champ d'en-tête Etag, contenant l'étiquette d'entité pour toutes les autres ressources du document, même si la ressource identifiée par la demande DELETE n'existe plus.

Quand un client utilise les opérations conditionnelles PUT et DELETE, il peut appliquer ces changements à la copie en antémémoire locale, et mettre à jour la valeur de l'étiquette d'entité pour la copie en antémémoire locale sur la base du champ d'en-tête Etag retourné dans la réponse. Tant qu'aucun autre client n'essaye de modifier le document, le client va être capable d'effectuer des opérations conditionnelles sur le document sans avoir jamais à effectuer d'opérations GET séparées pour synchroniser le document et ses étiquettes d'entité avec le serveur. Si un autre client essaye de modifier le document, cela va être détecté par les mécanismes conditionnels, et le client devra effectuer un GET pour resynchroniser sa copie sauf si il a d'autres moyens pour connaître le changement.

Si un client n'effectue pas d'opération conditionnelle, mais a une copie du document en antémémoire, cette copie va devenir invalide une fois l'opération effectuée (bien sûr, elle peut être devenue invalide même avant). Les opérations inconditionnelles ne devraient être effectuées par les clients que quand la connaissance du document entier n'est pas importante pour que l'opération réussisse.

Un autre exemple est quand un client va chercher un document, et qu'il y a une plus ancienne version en antémémoire, il est utile que les clients utilisent un GET conditionnel afin de réduire l'utilisation du réseau si la copie en antémémoire est encore valide. Ceci est fait en incluant, dans la demande GET, le champ d'en-tête If-None-Match avec une valeur égale à la etag actuelle détenue par le client pour le document. Le serveur va seulement générer une réponse 200 OK si la etag détenue par le serveur diffère de celle détenue par le client. Si elles ne diffèrent pas, le serveur va répondre avec une réponse 304.

8. Comportement du serveur

Un serveur XCAP est un serveur d'origine conforme à HTTP/1.1. Les comportements exigés par la présente spécification se rapportent à la façon dont l'URI HTTP est interprété et dont le contenu est construit.

Un serveur XCAP DOIT être explicitement au courant de l'usage d'application sur lequel les demandes sont faites. C'est-à-dire, le serveur doit être explicitement configuré à traiter les URI pour chaque usage d'application spécifique, et doit être au courant des contraintes imposées par cet usage d'application.

Quand le serveur reçoit une demande, le traitement dépend de l'URI. Si l'URI se réfère à un usage d'application non compris par le serveur, le serveur DOIT rejeter la demande avec une réponse 404 (Non trouvé). Si l'URI se réfère à un utilisateur (identifié par un XUI) qui n'est pas reconnu par le serveur, il DOIT rejeter la demande avec un 404 (Non trouvé). Si l'URI inclut des sélecteurs d'extension que le serveur ne comprend pas, il DOIT rejeter la demande avec un 404 (Non trouvé).

Ensuite, le serveur authentifie la demande. Tous les serveurs XCAP DOIVENT mettre en œuvre les résumé HTTP [RFC2617]. De plus, les serveurs DOIVENT mettre en œuvre HTTP sur TLS [RFC2818]. Il est RECOMMANDÉ que les administrateurs utilisent un URI HTTPS comme URI racine XCAP, afin que l'authentification du résumé du client se fasse sur TLS.

Ensuite, le serveur détermine si le client a l'autorisation d'effectuer l'opération demandée sur la ressource. L'usage d'application définit les politiques d'autorisation. Un usage d'application peut spécifier que la politique par défaut est utilisée. Cette politique par défaut est décrite au paragraphe 5.7.

Ensuite, le serveur s'assure qu'il peut correctement évaluer l'URI de demande. Le serveur DOIT séparer le sélecteur de document du sélecteur de nœud, en partageant l'URI à la première instance du séparateur de sélecteur de nœud ("~~"). Le serveur DOIT vérifier le sélecteur de nœud dans l'URI de demande, si il est présent. Si des noms qualifiés sont présents qui utilisent un préfixe d'espace de noms, et si ce préfixe n'est pas défini dans une expression xmlns() dans le composant d'interrogation de l'URI de demande, le serveur DOIT rejeter la demande avec une réponse 400.

Après la vérification des définitions de préfixe d'espace de noms, le comportement spécifique dépend de la méthode et de ce à quoi l'URI se réfère.

8.1 Traitement de POST

Les ressources XCAP ne représentent pas des descriptifs de traitement. Par suite, les opérations POST sur les URI HTTP qui représentent des ressources XCAP ne sont pas définies. Un serveur qui reçoit une telle demande pour une ressource XCAP DEVRAIT retourner une réponse 405.

8.2. Traitement de PUT

Le comportement d'un serveur à réception d'une demande PUT est comme spécifié dans HTTP/1.1, Section 9.6 -- le contenu de la demande est placé à la localisation spécifiée. Cette section sert à définir la notion de "placement" et de "localisation spécifiée" dans le contexte des ressources XCAP.

Si l'URI de demande contenait un sélecteur d'espace de noms, le serveur DOIT rejeter la demande avec une réponse 405 (Méthode non admise) et DOIT inclure un champ d'en-tête Allow incluant la méthode GET.

8.2.1 Localisation du parent

La première étape que le serveur effectue est de localiser le parent, qu'il soit un répertoire ou un élément, dans lequel la ressource est à placer. Pour ce faire, le serveur supprime le dernier segment de chemin de l'URI. Le reste de l'URI se réfère au parent. Ce parent peut être un document, un élément, ou un préfixe d'un sélecteur de document (appelé un répertoire, même si la présente spécification n'exige pas que les documents soient réellement mémorisés dans un système de fichiers). Cet URI est appelé l'URI parent. Le segment de chemin qui a été retiré est appelé le sélecteur de cible, et le nœud (élément, document, ou attribut) qu'il décrit est appelé le nœud cible.

Si l'URI parent n'a pas de séparateur de sélecteur de nœud, il se réfère au répertoire dans lequel le document devrait être inséré. Dans les opérations normales de XCAP, cela va être soit le répertoire d'accueil de l'utilisateur, soit le répertoire global, qui va toujours exister sur le serveur. Cependant, si un usage d'application utilise des sous répertoires (en dépit du fait que ce n'est pas recommandé) il est possible que le répertoire dans lequel le document devrait être inséré n'existe pas. Dans ce cas, le serveur DOIT retourner une réponse 409, et DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <no-parent>. Les rapports de conflit détaillés sont discutés à la Section 11. Si le répertoire existe, le serveur

vérifie si il y a un document avec le même nom de fichier que le nœud cible. Si il y en a un, l'opération est le remplacement, discuté au paragraphe 8.2.4. Si il n'existe pas, c'est l'opération de création discutée au paragraphe 8.2.3.

Si l'URI parent a un séparateur de sélecteur de nœud, le sélecteur de document est extrait, et ce document est restitué. Si le document n'existe pas, le serveur DOIT retourner une réponse 409, et DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <no-parent>. Si il existe bien, le sélecteur de nœud est extrait et décodé (on rappelle que le sélecteur de nœud est codé en pourcentage). Le sélecteur de nœud est appliqué au document sur la base des opérations de confrontation discutées au paragraphe 6.3. Si le résultat est une non correspondance ou est invalide, le serveur DOIT retourner une réponse 409, et DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <no-parent>.

Si le sélecteur de nœud est valide, le serveur examine le sélecteur de cible, et l'évalue dans le contexte du nœud parent. Si le nœud cible existe dans le parent, l'opération est un remplacement, comme décrit au paragraphe 8.2.4. Si il n'existe pas, c'est l'opération de création, discutée au paragraphe 8.2.3.

Avant d'effectuer le remplacement ou la création, comme déterminé sur la base de la logique ci-dessus, le serveur valide le contenu de la demande comme décrit au paragraphe 8.2.2.

8.2.2 Vérification du contenu du document

Si la demande PUT est pour un document (l'URI de demande n'a pas de séparateur de sélecteur de nœud) le contenu du corps de la demande doit être un document XML bien formé. Si il ne l'est pas, le serveur DOIT rejeter la demande avec un code de réponse 409. Cette réponse DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <not-well-formed>. Si le document est bien formé mais pas codé en UTF-8, le serveur DOIT rejeter la demande avec un code de réponse 409. Cette réponse DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <not-utf-8>. Si le type MIME dans le champ d'en-tête Type de contenu de la demande n'est pas égal au type MIME défini pour l'usage d'application, le serveur DOIT rejeter la demande avec un code 415.

Si la demande PUT est pour un élément, le contenu du corps de la demande doit être une région bien équilibrée d'un document XML, aussi appelée un corps de fragment XML dans la spécification d'échange de fragment XML [XML-Frag], incluant un seul élément. Si il ne l'est pas, le serveur DOIT rejeter la demande avec un code de réponse 409. Cette réponse DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <not-xml-frag>. Si le corps de fragment est bien équilibré mais contient des caractères en dehors du jeu de caractères UTF-8, le serveur DOIT rejeter la demande avec un code de réponse 409. Cette réponse DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <not-utf-8>. Si le type MIME dans le champ d'en-tête Type de contenu de la demande n'est pas égal à "application/xcap-el+xml", le serveur DOIT rejeter la demande avec un code 415.

Si la demande PUT est pour un attribut, le contenu du corps de la demande doit être une séquence de caractères qui se conforme à la grammaire pour AttValeur définie ci-dessus. Si il ne l'est pas, le serveur DOIT rejeter la demande avec un code de réponse 409. Cette réponse DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <not-xml-att-value>. Si la valeur de l'attribut est valide mais contient des caractères hors du jeu de caractères UTF-8, le serveur DOIT rejeter la demande avec un code de réponse 409. Cette réponse DEVRAIT inclure un rapport de conflit détaillé incluant l'élément <not-utf-8>. Si le type MIME dans le champ d'en-tête Type de contenu de la demande n'est pas égal à "application/xcap-att+xml", le serveur DOIT rejeter la demande avec un code 415.

8.2.3 Création

Les étapes de ce paragraphe sont suivies si la demande PUT résulte en la création d'un nouveau document, élément, ou attribut.

Si la demande PUT est pour un document, le contenu du corps de la demande est placé dans le répertoire, et son nom de fichier est associé au nœud cible, qui est un document.

Si la demande PUT est pour un élément, le serveur insère le contenu du corps de la demande comme nouvel élément fils de l'élément parent sélectionné au paragraphe 8.2.1. L'insertion est faite de telle sorte que l'URI de demande, quand il est évalué, pointe maintenant sur l'élément inséré. Il existe trois façons possibles de positionner ces nouveaux éléments.

D'abord, si il n'y avait pas d'autre élément apparenté avec le même nom expansé, et si l'insertion n'est pas contrainte positionnellement, le nouvel élément est inséré de telle façon qu'il soit le dernier élément parmi tous les éléments apparentés. De plus, si il y a des nœuds de commentaire, texte, ou instruction de traitement après l'ancien dernier élément,

ils DOIVENT se produire avant l'insertion du nouvel élément. Ce cas se produit quand une des conditions suivantes est vraie :

- Le nom d'élément dans le sélecteur de cible n'est pas muni d'un caractère générique. Ce pourrait être un sélecteur d'attribut (dans ce cas, il devrait correspondre à un attribut de l'élément inséré) et la position dans le sélecteur de cible va être absente ou avoir une valeur de 1 (une valeur supérieure à 1 résulterait toujours en le rejet de la demande, car c'est le premier élément avec le nom en question en dessous du parent).

- Le nom d'élément dans le sélecteur de cible est muni d'un caractère générique, mais il n'y a pas d'autre élément sous le même parent. Il pourrait y avoir un sélecteur d'attribut (dans ce cas, il devrait correspondre à un attribut de l'élément inséré) et la position dans le sélecteur de cible va être absente ou avoir une valeur de 1 (une valeur supérieure à 1 résulterait toujours en le rejet de la demande, car c'est le premier élément avec le nom en question en dessous du parent).

- Le nom d'élément dans le sélecteur de cible a un caractère générique, et il y a d'autres éléments sous le même parent. Cependant, il y a un sélecteur d'attribut qui ne correspond à aucun des attributs dans les autres éléments apparentés en dessous du parent, mais qui correspond à un attribut de l'élément à insérer. La position dans le sélecteur de cible est absente.

Ensuite, si il y avait déjà des éléments apparentés avec le même nom dans le document, mais si l'insertion n'est pas contrainte positionnellement, le serveur DOIT insérer l'élément de façon telle qu'il soit dans la position du "premier dernier". Le "premier dernier" signifie que le nouvel élément DOIT être inséré de telle façon qu'il 'y ait pas d'élément après lui avec le même nom expansé, et que pour toutes les positions d'insertion où cela est vrai, il soit inséré de telle façon que autant que possible de nœuds apparentés (élément, commentaire, texte, ou instruction de traitement) apparaissent après lui. Ce cas se produit quand le sélecteur de cible est défini par une production par nom ou par attribut, et qu'il n'y a pas de position indiquée.

Enfin, si l'élément est contraint positionnellement, le serveur DOIT insérer l'élément de façon qu'il soit dans la position du "énième plus tôt". Quand n>1 et que NameofAny n'est pas un caractère générique, l'élément DOIT être inséré de telle façon qu'il y aient n-1 éléments apparentés avant lui avec le même nom expansé. Si il n'y a pas n-1 éléments apparentés avec le même nom expansé, la demande va échouer. Quand n>1 et que NameorAny est un caractère générique, l'élément DOIT être inséré de façon à ce qu'il y aient n-1 éléments apparentés avant lui, chacun d'eux pouvant avoir tout nom expansé. Si il n'y a pas n-1 éléments apparentés dans le document, la demande va échouer. Dans ces deux cas, le nouvel élément est inséré de façon à ce que autant de nœuds apparentés que possible apparaissent après lui. Quand n=1 et que NameorAny n'est pas un caractère générique, l'insertion est contrainte positionnellement quand un élément avec le même nom expansé apparaît déjà comme fils du même parent. Dans ce cas, le nouvel élément DOIT apparaître juste avant le premier élément existant avec ce même nom expansé. Quand n=1 et que NameorAny est muni d'un caractère générique, l'insertion est contrainte positionnellement quand il y a aussi un sélecteur d'attribut qui ne correspond pas au premier apparenté du parent (si il ne correspondait pas, ou était absent, cela ne serait pas une insertion). Dans ce cas, le nouvel élément DOIT apparaître juste avant tous les éléments existants, sans considération de leur nom expansé.

En pratique, cette logique d'insertion garde étroitement ensemble les éléments avec les mêmes noms expansés. Cela simplifie la logique d'application quand le modèle de contenu est décrit par un schéma XML avec les règles de cardinalité <sequence> et maxOccurs="unbounded", comme :

```
<xs:nom d'élément="foobar">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="foo" maxOccurs="unbounded" />
      <xs:element ref="bar" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Sur la base de ce schéma, le document contient un certain nombre d'éléments <foo> suivis par un certain nombre d'éléments <bar>. Des éléments <bar> ou <foo> peuvent facilement être ajoutés sans caractère générique ni contrainte de position. Noter que si la cardinalité "minOccurs" de l'élément <foo> était zéro et si des éléments <foo> n'existent pas déjà, un prédicat positionnel avec le caractère générique * doit être utilisé.

La logique d'insertion complète est mieux décrite par des exemples complets. Considérons le document suivant :

```
<?xml version="1.0"?>
<root>
 <el1 att="first"/>
 <el1 att="second"/>
 <!-- commentaire -->
 <el2 att="first"/>
</root>
```

Une demande PUT dont le contenu est <el1 att="third"/> et dont le sélecteur de nœud est root/el1[@att="third"] résulterait en le document suivant :

```
<?xml version="1.0"?>
<root>
 <el1 att="first"/>
 <el1 att="second"/><el1 att="third"/>
 <!-- commentaire -->
 <el2 att="first"/>
</root>
```

On remarque comment il a été inséré comme troisième élément <el1> dans le document, et juste avant le commentaire et les nœuds d'espace. Il aurait été inséré exactement à la même place si le sélecteur de nœud avait été root/el1[3] [@att="third"] ou root/*[3][@att="third"].

Si le contenu de la demande avait été <el3 att="first"/> et si le sélecteur de nœud était root/el3, il aurait résulté en le document suivant :

```
<?xml version="1.0"?>
<root>
 <el1 att="first"/>
 <el1 att="second"/>
 <!-- commentaire -->
 <el2 att="first"/>
<el3 att="first"/></root>
```

Une demande PUT dont le contenu est <el2 att="2"/> et dont le sélecteur de nœud est root/el2[@att="2"] résulterait en le document suivant :

```
<?xml version="1.0"?>
<root>
 <el1 att="first"/>
 <el1 att="second"/>
 <!-- commentaire -->
 <el2 att="first"/><el2 att="2"/>
</root>
```

Il aurait été inséré exactement à la même place si le sélecteur de nœud avait été root/el2[2][@att="2"]. Cependant, un sélecteur root/*[2][@att="2"] aurait résulté en le document suivant :

```
<?xml version="1.0"?>
<root>
 <el1 att="first"/><el2 att="2"/>
 <el1 att="second"/>
 <!-- commentaire -->
 <el2 att="first"/>
</root>
```

Enfin, si le sélecteur de nœud avait été root/el2[1][@att="2"] le résultat serait :

```
<?xml version="1.0"?>
<root>
 <el1 att="first"/>
```

```
 <el1 att="second"/>
 <!-- commentaire -->
 <el2 att="2"/><el2 att="first"/>
</root>
```

Il est possible que l'élément ne puisse pas être inséré de façon que l'URI de demande, quand il est évalué, retourne le contenu fourni dans la demande. Une telle demande n'est pas permise pour PUT. Cela arrive quand l'élément dans le corps n'est pas décrit par l'expression dans le sélecteur de cible. Un exemple de ce cas est décrit au paragraphe 7.4. Si cela arrive, le serveur NE DOIT PAS effectuer l'insertion, et DOIT rejeter la demande avec une réponse 409. Le corps de la réponse DEVRAIT contenir un rapport de conflit détaillé contenant l'élément <ne-peut-pas-insérer>. Il est important de noter que la conformité de schéma ne joue pas de rôle dans l'insertion. C'est-à-dire, la décision sur l'endroit où l'élément est inséré est dictée entièrement par la structure de l'URI de demande, le document courant, et les règles de la présente spécification.

Si l'élément inséré (ou un de ses enfants) contient des déclarations d'espace de noms, ces déclarations sont conservées quand l'élément est inséré, même si ces mêmes déclarations existent dans un élément parent après l'insertion. Le serveur XCAP NE DOIT PAS retirer des déclarations d'espace de noms redondantes ou autrement changer les déclarations d'espace de noms qui étaient présentes dans l'élément inséré.

Si la demande PUT est pour un attribut, le serveur insère le contenu du corps de la demande comme valeur de l'attribut. Le nom de l'attribut est égal au att-name provenant du sélecteur d'attribut dans le sélecteur de cible.

En supposant que l'insertion peut être accomplie, le serveur vérifie que l'insertion résulte en un document qui satisfait aux contraintes de l'usage d'application. Ceci est présenté au paragraphe 8.2.5.

8.2.4 Remplacement

Les étapes de ce paragraphe sont suivies si la demande PUT va résulter en le remplacement d'un document, élément, ou attribut par le contenu de la demande.

Si la demande PUT est pour un document, le contenu du corps de la demande est placé dans le répertoire, remplaçant le document par le même nom de fichier.

Si la demande PUT est pour un élément, le serveur remplace le nœud cible par le contenu du corps de la demande. Comme dans le cas de création, il est possible que, après le remplacement, l'URI de demande ne choisisse pas l'élément qui vient d'être inséré. Si cela arrive, le serveur NE DOIT PAS effectuer le remplacement, et DOIT rejeter la demande avec une réponse 409. Le corps de la réponse DEVRAIT contenir un rapport de conflit détaillé contenant l'élément <ne-peut-pas-insérer>.

Comme avec la création, le remplacement d'un élément ne résulte pas en le changement ou l'élimination des déclarations d'espace de noms au sein de l'élément nouvellement modifié.

Si la demande PUT est pour un attribut, le serveur règle la valeur de l'attribut choisi au contenu du corps de la demande. Il est possible dans le cas du remplacement (mais pas dans le cas de création) qu'après le remplacement de l'attribut, l'URI de demande ne sélectionne plus l'attribut qui vient d'être remplacé. Le scénario dans lequel cela peut arriver est discuté au paragraphe 7.7. Si c'est le cas, le serveur NE DOIT PAS effectuer le remplacement, et DOIT rejeter la demande avec une réponse 409. Le corps de la réponse DEVRAIT contenir un rapport de conflit détaillé contenant l'élément <ne-peut-pas-insérer>.

8.2.5 Validation

Une fois qu'il a tenté d'insérer le document, élément, ou attribut, le serveur doit vérifier que le document résultant satisfait aux contraintes de données mentionnées par l'usage d'application.

D'abord, le serveur vérifie que le document final est conforme au schéma. Si il ne l'est pas, le serveur NE DOIT PAS effectuer l'insertion. Il DOIT rejeter la demande avec une réponse 409. Cette réponse DEVRAIT contenir un rapport de conflit détaillé contenant l'élément <schema-validation-error>. Si un schéma permet des éléments ou attributs provenant d'autres espaces de noms, et si le nouveau document contient des éléments ou attributs provenant d'un espace de noms inconnu, le serveur DOIT permettre le changement. En d'autres termes, il n'est pas nécessaire pour un serveur XCAP de comprendre les espaces de noms et les schémas correspondants pour les éléments et attributs dans un document, pour

autant que le schéma lui-même permette que de tels éléments ou attributs soient inclus. Bien sûr, de tels espaces de noms inconnus ne vont pas être annoncés par le serveur dans son document de capacités XCAP, discuté à la Section 12.

Si le document final contient des éléments ou attributs provenant d'un espace de noms que le serveur comprend (et a par conséquent annoncés dans son document de capacités XCAP) mais si le serveur n'a pas le schéma pour cet élément ou attribut particulier, le serveur DOIT rejeter la demande avec une réponse 409. Cette réponse DEVRAIT contenir un rapport de conflit détaillé contenant l'élément <schema-validation-error>.

Ensuite, le serveur vérifie toutes contraintes d'unicité identifiées par l'usage d'application. Si l'usage d'application exige qu'un élément ou attribut particulier ait une valeur unique dans une portée spécifique, le serveur va vérifier que cette propriété d'unicité existe encore. Si l'usage d'application exige qu'un URI dans le document soit unique dans le domaine, le serveur vérifie si c'est le cas. Si une de ces contraintes d'unicité n'est pas satisfaite, le serveur NE DOIT PAS effectuer l'insertion. Il DOIT rejeter la demande avec une réponse 409. Cette réponse DEVRAIT contenir un rapport de conflit détaillé contenant l'élément <uniqueness-failure>. Cet élément peut contenir des valeurs suggérées que le client peut utiliser pour réessayer. Ce DEVRAIT être des valeurs qui, au moment où le serveur génère le 409, vont satisfaire les contraintes d'unicité.

Le serveur vérifie aussi les contraintes d'URI et autres contraintes qui ne sont pas de schéma de données. Si le document échoue à une de ces contraintes, le serveur NE DOIT PAS effectuer l'insertion. Il DOIT rejeter la demande avec une réponse 409. Cette réponse DEVRAIT contenir un rapport de conflit détaillé contenant l'élément <constraint-failure>. Cet élément indique que le document a échoué à des contraintes qui ne sont pas de schéma de données explicitement invoquées par l'usage d'application.

Les suppressions d'élément ou d'attribut ont des contraintes similaires. Le serveur vérifie pour le document la validité du schéma et la conformité aux contraintes définies par l'usage d'application, et rejette la demande comme décrit ci-dessus, si l'une ou l'autre des vérifications échoue.

8.2.6 Traitement conditionnel

Une demande PUT pour une ressource XCAP, comme toute autre ressource HTTP, peut être rendue conditionnelle par l'usage des champs d'en-tête If-Match et If-None-Match. Pour un remplacement, ils sont traités comme défini dans la [RFC2616]. Pour une insertion d'un élément ou attribut, les opérations conditionnelles sont permises. L'étiquette d'entité qui est utilisée pour les procédures de la [RFC2616] est celle pour toutes les ressources dans le même document que le parent de l'élément ou attribut inséré. Une façon de voir cela est que, d'un point de vue logique, à réception de la demande PUT, le serveur XCAP instancie la etag pour la ressource référencée par la demande, et ensuite applique le traitement de la demande. À cause de ce comportement, il n'est pas possible d'effectuer une insertion conditionnelle sur un attribut ou élément qui est conditionnel sur l'opération qui est une insertion et non un remplacement. En d'autres termes, un PUT conditionnel d'un élément ou attribut avec un If-None-Match: * va toujours échouer.

8.2.7 Interdépendances de ressources

Parce que les ressources XCAP incluent des éléments, attributs, et documents, dont chacun a son propre URI HTTP, la création ou modification d'une ressource affecte l'état de beaucoup d'autres. Par exemple, l'insertion d'un document crée des ressources sur le serveur pour tous les éléments et attributs au sein de ce document. Après que le serveur a effectué l'insertion associée au PUT, le serveur DOIT créer et/ou modifier ces ressources affectées par le PUT. La structure du document définit complètement les inter relations entre ces ressources.

Cependant, l'usage d'application peut spécifier d'autres inter dépendances de ressources. Le serveur DOIT créer ou modifier les ressources spécifiées par l'usage d'application.

Si la création ou remplacement a réussi, et si les interdépendances de ressources sont résolues, le serveur retourne respectivement un 201 Créé ou un 200 OK. Noter qu'un 201 Créé est généré pour la création de nouveaux documents, éléments, ou attributs. Une réponse 200 OK à un PUT NE DOIT PAS avoir de contenu. Selon les recommandations de la RFC 2616, le code 201 peut contenir un champ d'en-tête Localisation et l'entité qui identifie la ressource qui a été créée. Une étiquette d'entité DOIT être incluse dans toutes les réponses de succès à un PUT.

8.3 Traitement de GET

La sémantique de GET est comme spécifié dans la RFC 2616. Ce paragraphe précise le contenu spécifique à retourner pour

un URI particulier qui représente une ressource XCAP.

Si l'URI de demande contient seulement un sélecteur de document, le serveur retourne le document spécifié par l'URI si il existe, et autrement il retourne une réponse 404. Le type MIME du corps de la réponse 200 OK DOIT être le type MIME défini par cet usage d'application (c'est-à-dire, "application/resource-lists+xml").

Si l'URI de demande contient un sélecteur de nœud, le serveur obtient le document spécifié par le sélecteur de document, et si il est trouvé, évalue le sélecteur de nœud au sein de ce document. Si aucun document n'est trouvé, ou si le sélecteur de nœud est une non correspondance ou est invalide, le serveur retourne une réponse 404. Autrement, le serveur retourne une réponse 200 OK. Si le sélecteur de nœud identifie un élément XML, cet élément est retourné dans la réponse 200 OK comme un corps de fragment XML contenant l'élément retenu. Le serveur NE DOIT PAS ajouter de liens d'espace de noms représentant les espaces de noms utilisés par l'élément ou ses enfants, mais déclarés dans des éléments ancêtres ; le client va déjà connaître ces liens (car il a une copie en antémémoire du document entier) ou il peut les apprendre en interrogeant explicitement sur les liens. Le type MIME de la réponse DOIT être "application/xcap-el+xml". Si le sélecteur de nœud identifie un attribut XML, la valeur de cet attribut est retournée dans le corps de la réponse. Le type MIME de la réponse DOIT être "application/xcap-att+xml". Si le sélecteur de nœud identifie un ensemble de liens d'espace de noms, le serveur calcule l'ensemble des liens d'espace de noms dans la portée pour l'élément (incluant celui par défaut) et le code en utilisant le format "application/xcap-ns+xml" défini à la Section 10. Ce document est ensuite retourné dans le corps de la réponse.

Les opérations GET peuvent être conditionnelles, et suivre les procédures définies dans la [RFC2616].

Noter que le GET d'une ressource qui a juste fait l'objet d'un PUT peut n'être pas l'équivalent octet par octet de ce qu'était le PUT, à cause de la normalisation XML et des règles d'équivalence.

Une réponse de réussite à un GET DOIT inclure une étiquette d'entité.

8.4 Traitement de DELETE

La sémantique de DELETE est comme spécifié dans la RFC 2616. Ce paragraphe précise le contenu spécifique à supprimer pour un URI particulier qui représente une ressource XCAP.

Si l'URI de demande contenait un sélecteur d'espace de noms, le serveur DOIT rejeter la demande avec un code 405 (Méthode non admise) et DOIT inclure un champ d'en-tête Allow incluant la méthode GET.

Si l'URI de demande contient seulement un sélecteur de document, le serveur supprime le document spécifié par l'URI si il existe et retourne un 200 OK ; autrement, il retourne une réponse 404.

Si l'URI de demande contient un sélecteur de nœud, le serveur obtient le document spécifié par le sélecteur de document, et si il est trouvé, évalue le sélecteur de nœud dans ce document. Si aucun document n'est trouvé, ou si le sélecteur de nœud est une non correspondance ou est invalide (noter qu'il va être invalide si plusieurs éléments ou attributs sont sélectionnés) le serveur retourne une réponse 404. Autrement, le serveur supprime l'élément ou attribut spécifié du document et effectue la vérification de validation définie au paragraphe 8.2.5. Noter que cette suppression n'inclut aucune espace autour de l'élément qui est supprimé ; le serveur XCAP DOIT préserver les espaces entourantes. Il est possible que, après la suppression, l'URI de demande choisisse un autre élément dans le document. Si cela arrive, le serveur NE DOIT PAS effectuer la suppression, et DOIT rejeter la demande avec une réponse 409. Le corps de la réponse DEVRAIT contenir un rapport de conflit détaillé contenant l'élément <ne-peut-pas-supprimer>. Si la suppression va causer une défaillance à une des contraintes, la suppression NE DOIT PAS avoir lieu. Le serveur suit les procédures du paragraphe 8.2.5 pour calculer la réponse 409. Si la suppression résulte en un document qui est encore valide, le serveur DOIT effectuer la suppression, traiter les interdépendances de ressources définies par l'usage d'application, et retourner une réponse 200 OK.

Les opérations DELETE peuvent être conditionnelles, et suivent les procédures définies dans la [RFC2616].

Avant que le serveur retourne la réponse 200 OK à un DELETE, il DOIT traiter les interdépendances de ressources comme défini au paragraphe 8.2.7. Tant que le document subiste après l'opération de suppression, toute réponse de réussite à DELETE DOIT inclure l'étiquette d'entité du document.

8.5 Gestion des étiquettes d'entité

Un serveur XCAP DOIT conserver les étiquettes d'entité pour toutes les ressources qu'il maintient. La présente

spécification introduit la contrainte suppémentaire que quand une ressource dans un document (incluant le document lui-même) change, cette ressource reçoit une nouvelle etag, et toutes les autres ressources dans ce document DOIVENT recevoir la même valeur de etag. Effectivement, il y a une seule etag pour le document entier. Un serveur XCAP DOIT inclure le champ d'en-tête Etag dans toutes les réponses 200 ou 201 à PUT, GET, et DELETE, en supposant que le document lui-même existe encore après l'opération. Dans le cas d'un DELETE, l'étiquette d'entité se réfère à la valeur de l'étiquette d'entité pour le document après la suppression de l'élément ou attribut.

Les ressources XCAP n'introduisent pas de nouvelles exigences sur la force des étiquettes d'entité.

Par suite de cette contrainte, quand un client fait un changement à un élément ou attribut au sein d'un document, la réponse à cette opération va porter l'étiquette d'entité de la ressource qui vient d'être affectée. Comme le client saît que cette valeur d'étiquette d'entité est partagée par toutes les autres ressources dans le document, le client peut faire des demandes conditionnelles sur les autres ressources en utilisant cette étiquette d'entité.

9. Contrôle d'antémémoire

Une ressource XCAP est une ressource HTTP valide, et donc, elle peut être mise en antémémoire par les clients et les antémémoires du réseau. Les antémémoires du réseau, ne vont cependant pas être conscientes des interdépendances entre les ressources XCAP. À ce titre, un changement sur un élément dans un document par un client va invalider d'autres ressources XCAP affectées par le changement. Pour les usages d'application qui contiennent des données qui sont probablement dynamiques ou écrites par les clients, les serveurs DEVRAIENT indiquer une directive «no-cache».

10. Format de lien d'espace de noms

Un sélecteur de nœud peut identifier un ensemble de liens d'espace de noms qui sont dans la portée pour un élément particulier. Afin de convoyer ces liens dans une réponse GET, il y a besoin d'un moyen pour les coder.

Le codage est facilement fait en incluant un seule élément XML dans un corps de fragment XML. Cet élément a le même nom local que l'élément dont les liens d'espace de noms sont désirés, et aussi le même préfixe d'espace de noms. L'élément a un attribut xmlns qui identifie l'espace de noms par défaut dans la portée, et une déclaration xmlns:prefix pour chaque préfixe dans la portée.

Par exemple, considérons le document XML du paragraphe 6.4. Le sélecteur de nœud df:foo/df2:bar/df2:baz/namespace::* va retenir les espaces de noms dans la portée pour l'élément <baz> dans le document, en supposant que la demande est accompagnée par un composant d'interrogation qui contient xmlns(df=urn:test:default-namespace) et xmlns(df2=urn:test:namespace1-uri). Un GET contenant ce sélecteur de nœud et ces liens d'espace de noms va produire le résultat suivant :

```
<baz xmlns="urn:test:namespace1-uri"
    xmlns:ns1="urn:tes:namespace1-uri"/>
```

Il est important de noter que le client n'a pas besoin de savoir les liens d'espace de noms réels afin de construire l'URI. Il a besoin de savoir l'URI d'espace de noms pour chaque élément dans le sélecteur de nœud. Les liens d'espace de noms présents dans le composant d'interrogation sont définis par le client, transposant ces URI en un ensemble de préfixes. Les liens retournés par le serveur sont les liens réels utilisés dans le document.

11. Rapports de conflit détaillés

Dans les cas où le serveur retourne une réponse d'erreur 409, cette réponse va généralement inclure un document dans le corps de la réponse qui donne plus de détails sur la nature de l'erreur. Le présent document est un document XML, formaté conformément au schéma du paragraphe 11.2. Son type MIME, enregistré par cette spécification, est "application/xcap-error+xml".

11.1 Structure de document

La structure de document est simple. Elle contient l'élément racine <xcap-error>. Le contenu de cet élément est une condition d'erreur spécifique. Chaque condition d'erreur est représentée par un élément différent. Cela permet que différentes conditions d'erreur fournissent des données différentes sur la nature de l'erreur. Tous les éléments d'erreur prennent un attribut "phrase", qui peut contenir du texte destiné à l'utilisateur humain.

Les éléments d'erreur suivants sont définis dans la présente spécification :

<pas-bien-formé> : cela indique que le corps de la demande n'était pas un document XML bien formé.

<pas-frag-xml> : cela indique que la demande était supposée contenir un corps de fragment XML valide, mais ce n'est pas le cas. Très probablement parce que le XML dans le corps est mal formé ou pas équilibré.

<pas-de-parent> : cela indique qu'une tentative d'insertion d'un document, élément, ou attribut a échoué parce que le répertoire, document, ou élément dans lequel l'insertion était supposée se produire n'existe pas. Cet élément d'erreur peut contenir un élément facultatif <ancestor>, qui donne un URI HTTP représentant le plus proche parent qui serait un point d'insertion valide. Cet URI HTTP PEUT être un URI relatif, par rapport au document lui-même. Parce que c'est un URI HTTP valide, son composant de sélecteur de nœud DOIT être codé en pourcentage.

<schema-validation-error> : cet élément indique que le document n'était pas conforme au schéma après l'opération demandée.

<pas-valeur-att-xml>: cela indique que la demande était supposée contenir une valeur valide d'attribut XML, mais ce n'est pas le cas.

<ne-peut-pas-insérer> : cela indique que l'opération PUT demandée n'a pas pu être effectuée parce que un GET de cette ressource après le PUT ne va pas donner le contenu de la demande PUT.

<ne-peut-pas-supprimer> : cela indique que l'opération DELETE demandée n'a pas pu être effectuée parce que elle ne serait pas idempotente.

<échec-d'unicité> : cela indique que l'opération demandée résulterait en un document qui ne satisferait pas la contrainte d'unicité définie par l'usage d'application. Pour chaque URI, élément, ou attribut spécifié par le client qui n'est pas unique, un élément <existe> est présent comme contenu de l'élément d'erreur. Chaque élément <existe> a un attribut "field" qui contient un URI relatif identifiant l'élément ou attribut XML dont la valeur a besoin d'être unique, mais ne l'était pas. L'URI relatif est relatif au document lui-même, et va donc commencer à l'élément racine. Le composant d'interrogation de l'URI DOIT être présent si la portion sélecteur de nœud de l'URI contient des préfixes d'espace de noms. Comme le sélecteur de nœud "field" est un URI HTTP valide, il DOIT être codé en pourcentage. L'élément <existe> peut facultativement contenir une liste d'éléments <alt-value>. Chacun est une valeur de remplacement suggérée qui n'existe actuellement pas sur le serveur.

<contrainte-défaillante> : cela indique que l'opération demandée résulterait en un document qui ne respecterait pas une contrainte de données définie par l'usage d'application, mais non appliquée par le schéma, ou une contrainte d'unicité.

<extension> : cela indique une condition d'erreur qui est définie par une extension à XCAP. Les clients qui ne comprennent pas le contenu de l'élément d'extension DOIVENT éliminer le document xcap-error et traiter l'erreur comme un 409 non qualifié.

<pas-utf-8> : cela indique que la demande n'a pas pu être achevée parce que elle aurait produit un document non codé en UTF-8.

Par exemple, le document suivant indique que l'utilisateur a tenté de créer un service RLS en utilisant l'URI sip:amis@exemple.com, mais cet URI existe déjà :

```
<?xml version="1.0" encoding="UTF-8"?>
<xcap-error xmlns="urn:ietf:params:xml:ns:xcap-error">
 <échec-d'unicité>
  <champ existe="rls-services/service/@uri">
   <alt-value>sip:mespotes@exemple.com</alt-value>
  </existe>
```

```
 </échec-d'unicité>
</xcap-error>
```

11.2 Schéma XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:xcap-error"
 xmlns="urn:ietf:params:xml:ns:xcap-error"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">

 <xs:nom d'élément="error-element" abstract="true"/>
 <xs:nom d'élément="xcap-error">
  <xs:annotation>
   <xs:documentation>Indique la raison de l'erreur.
  </xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="error-element"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:nom d'élément="extension" substitutionGroup="error-element">
  <xs:complexType>
   <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:nom d'élément="schema-validation-error" substitutionGroup="error-element">
  <xs:annotation>
   <xs:documentation>Cet élément indique que le document est non conforme au schéma après l'opération
     demandée.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:attribute name="phrase" type="xs:string" use="optional"/>
  </xs:complexType>
 </xs:element>

 <xs:nom d'élément="not-xml-frag" substitutionGroup="error-element">
  <xs:annotation>
   <xs:documentation>cela indique que la demande était supposée contenir un corps de fragment XML valide, mais ce n'est
     pas le cas.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:attribute name="phrase" type="xs:string" use="optional"/>
  </xs:complexType>
 </xs:element>

 <xs:nom d'élément="no-parent" substitutionGroup="error-element">
  <xs:annotation>
   <xs:documentation>cela indique qu'une tentative d'insertion d'un élément, attribut, ou document a échoué parce que le
     document ou élément dans lequel l'insertion était supposée se produire n'existe pas.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
```

```
 <xs:nom d'élément="ancestor" type="xs:anyURI" minOccurs="0">
  <xs:annotation>
   <xs:documentation>Contient un URI HTTP qui pointe sur l'élément qui est le plus proche ancêtre existant.
   </xs:documentation>
  </xs:annotation>
 </xs:element>
</xs:sequence>
<xs:attribute name="phrase" type="xs:string" use="optional"/>
/xs:complexType>
</xs:element>

 <xs:nom d'élément="ne-peut-pas-insérer" substitutionGroup="error-element">
  <xs:annotation>
   <xs:documentation>cela indique que l'opération PUT demandée n'a pas pu être effectuée parce que un GET de cette
     ressource après le PUT ne donne pas le contenu de la demande PUT.
   </xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:attribute name="phrase" type="xs:string" use="optional"/>
  </xs:complexType>
 </xs:element>

 <xs:nom d'élément="pas-valeur-att-xml"
  substitutionGroup="error-element">
  <xs:annotation>
   <xs:documentation>cela indique que la demande était supposée contenir une valeur valide d'attribut XML, mais ce n'est
     pas le cas.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:attribute name="phrase" type="xs:string" use="optional"/>
  </xs:complexType>
 </xs:element>

 <xs:nom d'élément="uniqueness-failure"
  substitutionGroup="error-element">
  <xs:annotation>
   <xs:documentation>cela indique que l'opération demandée résulterait en un document qui ne satisferait pas à une
     contrainte d'unicité définie par l'usage d'application.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:nom d'élément="existe" maxOccurs="unbounded">
     <xs:annotation>
      <xs:documentation>Pour chaque URI, élément, ou attribut spécifié par le client qui n'est pas unique, un d'eux est
          présent.</xs:documentation>
     </xs:annotation>
     <xs:complexType>
      <xs:sequence minOccurs="0">
       <xs:nom d'élément="alt-value" type="xs:string"
        maxOccurs="unbounded">
        <xs:annotation>
         <xs:documentation>Un ensemble facultatif de valeurs de remplacement peut être fourni.</xs:documentation>
        </xs:annotation>
       </xs:element>
      </xs:sequence>
      <xs:attribute name="field" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
   <xs:attribute name="phrase" type="xs:string" use="optional"/>
```

```
 </xs:complexType>
</xs:element>

<xs:nom d'élément="pas-bien-formé"
 substitutionGroup="error-element">
 <xs:annotation>
  <xs:documentation>cela indique que le corps de la demande n'est pas un document bien formé.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:attribute name="phrase" type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>

<xs:nom d'élément="constraint-failure"
 substitutionGroup="error-element">
 <xs:annotation>
  <xs:documentation>Cela indique que l'opération demandée résulterait en un document qui ne respecte pas une contrainte
de données définie par l'usage d'application, mais non appliquée par le schéma, ou une contrainte d'unicité.
</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:attribute name="phrase" type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>

<xs:nom d'élément="ne-peut-pas-supprimer" substitutionGroup="error-element">
 <xs:annotation>
  <xs:documentation>Cela indique que l'opération DELETE demandée n'a pas pu être effectuée parce que elle ne serait pas
    idempotente.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:attribute name="phrase" type="xs:string" use="optional"/>
 </xs:complexType>
  </xs:element>

<xs:nom d'élément="pas-utf-8" substitutionGroup="error-element">
 <xs:annotation>
  <xs:documentation>Cela indique que la demande n'a pas pu être achevée parce que elle aurait produit un document non
    codé en UTF-8.</xs:documentation>
 </xs:annotation>
 <xs:complexType>
  <xs:attribute name="phrase" type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>
</xs:schema>
```

12. Capacités de serveur XCAP

XCAP peut être étendu par l'ajout de nouveaux usages d'application et extensions au cœur de protocole. Les usages d'application peuvent définir des types MIME avec des schémas XML qui permettent de nouveaux nœuds d'extension à partir de nouveaux espaces de noms. Il sera souvent nécessaire qu'un client détermine quelles extensions, usages d'application, ou espaces de noms un serveur prend en charge avant de faire une demande. Pour permettre cela, la présente spécification définit un usage d'application avec l'AUID "xcap-caps". Tous les serveurs XCAP DOIVENT prendre en charge cet usage d'application. Cet usage définit un seul document au sein de l'arborescence globale qui fait la liste des capacités du serveur. Les clients peuvent lire ce document bien connu, et donc apprendre les capacités du serveur.

La structure du document est simple. L'élément racine est <xcap-caps>. Ses enfants sont <auids>, <extensions>, et <namespaces>. Chacun d'eux contient une liste des AUID, extensions, et espaces de noms pris en charge par le serveur. Les extensions sont désignées par des jetons définis par l'extension, et définissent normalement de nouveaux sélecteurs.

Les espaces de noms sont identifiés par leur URI d'espace de noms. Pour 'prendre en charge' un espace de noms, le serveur doit avoir les schémas pour tous les éléments au sein de cet espace de noms, et être capable de les valider si ils apparaissent au sein des documents. Comme tous les serveurs XCAP prennent en charge l'AUID "xcap-caps", il DOIT figurer dans l'élément <auids>, et l'espace de noms "urn:ietf:params:xml:ns:xcap-caps" DOIT figurer dans l'élément <namespaces>.

Les paragraphes qui suivent donnent les informations nécessaires pour définir cet usage d'application.

12.1 Identifiant unique d'application (AUID)

La présente spécification définit l'AUID "xcap-caps" au sein de l'arborescence de l'IETF, via l'enregistrement de l'IANA à la Section 15.

12.2 Schéma XML

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema targetNamespace="urn:ietf:params:xml:ns:xcap-caps"
   xmlns="urn:ietf:params:xml:ns:xcap-caps"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:nom d'élément="xcap-caps">
    <xs:annotation>
     <xs:documentation>Élément racine pour xcap-caps</xs:documentation>
    </xs:annotation>
    <xs:complexType>
     <xs:sequence>
      <xs:nom d'élément="auids">
       <xs:annotation>
        <xs:documentation>Liste des AUID pris en charge.</xs:documentation>
       </xs:annotation>
       <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:nom d'élément="auid" type="auidType"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:nom d'élément="extensions" minOccurs="0">
       <xs:annotation>
        <xs:documentation>Liste des extensions prises en charge.
        </xs:documentation>
       </xs:annotation>
       <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:nom d'élément="extension" type="extensionType"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:nom d'élément="namespaces">
       <xs:annotation>
        <xs:documentation>Liste des espaces de noms pris en charge.
        </xs:documentation>
       </xs:annotation>
       <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:nom d'élément="namespace" type="namespaceType"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:any namespace="##other" processContents="lax"
       minOccurs="0" maxOccurs="unbounded"/>
```

```
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:simpleType name="auidType">
   <xs:annotation>
    <xs:documentation>Type d'AUID</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:simpleType name="extensionType">
   <xs:annotation>
    <xs:documentation>Type d'extension</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:simpleType name="namespaceType">
   <xs:annotation>
    <xs:documentation>Type d'espace de noms</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:anyURI"/>
  </xs:simpleType>
 </xs:schema>
```

12.3. Espace de noms de document par défaut

L'espace de noms de document par défaut utilisé pour évaluer un URI est urn:ietf:params:xml:ns:xcap-caps.

12.4 Type MIME

Les documents conformes à ce schéma sont connus par le type MIME "application/xcap-caps+xml", enregistré au paragraphe 15.2.5.

12.5 Contraintes de validation

Il n'y a pas de contrainte de validation supplémentaire associée à cet usage d'application.

12.6 Sémantique des données

La sémantique des données est définie ci-dessus.

12.7 Conventions de dénomination

Un serveur DOIT conserver une seule instance du document dans l'arborescence globale, en utilisant le nom de fichier "index". Il NE DOIT PAS y avoir d'instance de ce document dans l'arborescence de l'utilisateur.

12.8 Interdépendances de ressources

Il n'y a pas d'interdépendance de ressources dans cet usage d'application au delà de celles définies par le schéma.

12.9 Politiques d'autorisation

Cet usage d'application ne change pas la politique d'autorisation par défaut définie par XCAP.

13. Exemples

Cette section donne plusieurs exemples, en utilisant les usages d'application XCAP de listes de ressource et de services de RLS [RFC4826].

D'abord, un utilisateur Bill crée un nouveau document (voir le paragraphe 7.1). Le présent document est une nouvelle liste de ressources, initialement avec une seule liste, appelée "amis", sans utilisateur dedans :

```
PUT
/resource-lists/users/sip:bill@exemple.com/index HTTP/1.1
Content-Type:application/resource-lists+xml
Host: xcap.exemple.com

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
  <list name="amis">
  </list>
</resource-lists>
```

Figure 24 : Nouveau document

Ensuite, Bill crée un document de services RLS définissant un seul service RLS faisant référence à cette liste. Ce service a un URI de sip:mesamis@exemple.com (les URI reviennent à la ligne pour la lisibilité) :

```
PUT
/rls-services/users/sip:bill@exemple.com/index HTTP/1.1
Content-Type:application/rls-services+xml
Host: xcap.exemple.com

<?xml version="1.0" encoding="UTF-8"?>
<rls-services xmlns="urn:ietf:params:xml:ns:rls-services">
<service uri="sip:mesamis@exemple.com">
  <resource-list>http://xcap.exemple.com/resource-lists/users/
sip:bill@exemple.com/index/~~/resource-lists/
list%5b@name=%22amis%22%5d
</resource-list>
  <packages>
   <package>presence</package>
   </packages>
 </service>
</rls-services>
```

Figure 25 : Exemple de services de RLS

Ensuite, Bill crée un élément dans le document de listes de ressources (paragraphe 7.4). En particulier, il ajoute une entrée à la liste :

```
PUT
/resource-lists/users/sip:bill@exemple.com/index
/~~/resource-lists/list%5b@name=%22amis%22%5d/entry HTTP/1.1
Content-Type:application/xcap-el+xml
Host: xcap.exemple.com

<entry uri="sip:bob@exemple.com">
    <display-name>Bob Jones</display-name>
  </entry>
```

Figure 26 : Document de listes de ressources

Ensuite, Bill va chercher le document (paragraphe 7.3):

```
GET
/resource-lists/users/sip:bill@exemple.com/index HTTP/1.1
```

Figure 27 : Opération Fetch

Et le résultat est (noter comment les nœuds de texte d'espace apparaissent dans le document):

```
HTTP/1.1 200 OK
Etag: "wwhha"
Content-Type: application/resource-lists+xml

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
  <list name="amis">
  <entry uri="sip:bob@exemple.com">
    <display-name>Bob Jones</display-name>
  </entry></list>
</resource-lists>
```

Figure 28 : Résultat de Fetch

Ensuite, Bill ajoute une autre entrée à la liste, qui est une autre liste avec trois entrées. C'est une autre création d'éléments (paragraphe 7.4):

```
PUT
/resource-lists/users/sip:bill@exemple.com/index/~~/resource-lists/list%5b@name=%22amis%22%5d/
list%5b@name=%22amis-proches%22%5d HTTP/1.1
Content-Type: application/xcap-el+xml
Host: xcap.exemple.com

<list name="amis-proches">
   <entry uri="sip:joe@exemple.com">
     <display-name>Joe Smith</display-name>
   </entry>
   <entry uri="sip:nancy@exemple.com">
     <display-name>Nancy Gross</display-name>
   </entry>
   <entry uri="sip:petri@exemple.com">
     <display-name>Petri Aukia</display-name>
   </entry>
</list>
```

Figure 29 : Ajout d'une entrée

Puis, Bill décide qu'il ne veut pas Petri sur la liste, donc il supprime l'entrée (paragaphe 7.5):

```
DELETE
/resource-lists/users/sip:bill@exemple.com/index/~~/resource-lists/list/list/
entry%5b@uri=%22sip:petri@exemple.com%22%5d HTTP/1.1
Host: xcap.exemple.com
```

Figure 30 : Suppression d'une entrée

Bill décide de vérifier l'URI pour Nancy, donc il va chercher un attribut particulier (paragraphe 7.6):

```
GET
/resource-lists/users/sip:bill@exemple.com/index/~~/resource-lists/list/list/entry%5b2%5d/@uri HTTP/1.1
Host: xcap.exemple.com
```

Figure 31 : Aller chercher un attribut

et le serveur répond :

HTTP/1.1 200 OK
Etag: "ad88"
Content-Type:application/xcap-att+xml
"sip:nancy@exemple.com"

Figure 32 : Résultat de Fetch

14. Considérations sur la sécurité

Fréquemment, les données manipulées par XCAP contiennent des informations sensibles. Pour éviter que des espions voient ces informations, il est RECOMMANDÉ qu'un administrateur établisse un URI HTTPS comme URI racine XCAP. Il va en résulter des communications chiffrées par TLS entre le client et le serveur, empêchant tout espionnage. Les clients DOIVENT mettre en œuvre TLS, assurant que de tels URI seront utilisables par le client.

L'authentification du client et du serveur est aussi importante. Un client a besoin d'être sûr qu'il parle au serveur qu'il croit qu'il a contacté. Autrement, il peut lui donner de fausses informations, qui peuvent conduire à des attaques de déni de service contre un client. Pour empêcher cela, un client DEVRAIT tenter de mettre à niveau [RFC2817] toutes les connexions avec TLS. De même, l'autorisation des opérations de lecture et écriture sur les données est importante, et cela exige l'authentification du client. Par suite, un serveur DEVRAIT mettre au défi un client en utilisant le résumé HTTP [RFC2617] pour établir son identité, et cela DEVRAIT être fait sur une connexion TLS. Les clients DOIVENT mettre en œuvre l'authentification par résumé, assurant l'interopérabilité avec les serveurs qui mettent au défi le client. Les serveurs NE DOIT PAS effectuer l'authentification de base sans une connexion TLS avec le client.

Parce que XCAP est un usage de HTTP et non un protocole séparé, il fonctionne sur le même numéro d'accès que le trafic HTTP normal. Cela rend difficele l'application des règles de filtrage fondées su l'accès dans les pare-feu pour séparer le traitement du trafic XCAP des autres trafics HTTP.

Cependant, ce problème existe aujourd'hui largement parce que HTTP est utilisé pour accéder à une grande diversité de contenus, tous sur le même accès, et XCAP voit les documents de configuration d'application comme juste un autre type de contenu HTTP. À ce titre, un traitement séparé du trafic XCAP des autres trafics HTTP exige que les pare-feu examinent l'URL lui-même. Il n'y a pas de moyen à toute épreuve pour identifier un URL comme pointant sur une ressource XCAP. Cependant, la présence du double tilde (~~) est une indication forte de ce que l'URL pointe sur un élément ou attribut XML. Comme toujours, il faut faire attention à chercher le double-tilde à cause de toutes les façons dont un URI peut être codé sur le réseau [RFC3986], [RFC3987].

15. Considérations relatives à l'IANA

Plusieurs considérations relatives à l'IANA sont associées à la présente spécification.

15.1 Identifiants uniques d'application XCAP

Conformément aux instructions de cette spécification, l'IANA a créé un nouveau registre pour les identifiants uniques d'application XCAP (AUID). Ce registre est défini comme un tableau contenant trois colonnes :

AUID : c'est une chaîne fournie dans les enregistrements de l'IANA dans le registre.
Description : c'est le texte fourni par l'enregistrement de l'IANA dans le registre
Référence : c'est une référence à la RFC qui contient l'enregistrement.

Conformément aux instructions de cette spécification, l'IANA a créé ce tableau avec une entrée initiale. Le tableau résultant ressemble à :

Identifiant unique d'application (AUID)	**Description**	**Référence**
xcap-caps	Capacités d'un serveur XCAP	RFC 4825

Les AUID XCAP sont enregistrés par l'IANA quand ils sont publiés dans des RFC sur la voie de la normalisation. La Section Considérations relatives à l'IANA de la RFC doit inclure les informations suivantes, qui apparaissent dans le registre IANA avec le numéro de la RFC publiée.

- Nom de l'AUID. Le nom PEUT être d'une longueur quelconque, mais NE DEVRAIT PAS faire plus de 20 caractères. Le nom DOIT consister seulement en caractères alphanumériques et de marquage [RFC3261].

- Texte descriptif de l'usage d'application.

15.2 Types MIME

La présente spécification demande l'enregistrement de plusieurs nouveaux types MIME conformément aux procédures de la [RFC4288] et des lignes directrices de la [RFC3023].

15.2.1 Type MIME application/xcap-el+xml

Nom de type de support MIME : application
Nom de sous type MIME : xcap-el+xml
Paramètres obligatoires : aucun
Paramètres facultatifs : les mêmes que le paramètre de jeu de caractère application/xml comme spécifié dans la [RFC3023].
Considérations de codage : les mêmes que les considérations de codage de application/xml comme spécifié dans la [RFC3023].
Considérations de sécurité : voir la Section 10 de la [RFC3023].
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 4825
Applications qui utilisent ce type de supports : le présent type de document a été utilisé pour prendre en charge le transport de corps de fragment XML dans la RFC 4825, le protocole d'accès à la configuration XML (XCAP).

Informations supplémentaires :
Numéro magique : aucun
Extension de fichier : .xel
Code de type de fichier Macintosh : "TEXT"
Adresse personnelle et de messagerie pour plus d'informations : Jonathan Rosenberg, jdrosen@jdrosen.net
Utilisation prévue : COURANTE
Auteur/contrôleur des changements : l'IETF.

15.2.2 Type MIME application/xcap-att+xml

Nom de type de support MIME : application
Nom de sous type MIME : xcap-att+xml
Paramètres obligatoires : aucun
Paramètres facultatifs : les mêmes que le paramètre de jeu de caractère application/xml comme spécifié dans la [RFC3023].
Considérations de codage : les mêmes que les considérations de codage de application/xml comme spécifié dans la [RFC3023].
Considérations de sécurité : voir la Section 10 de la [RFC3023].
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 4825
Applications qui utilisent ce type de supports : le présent type de document a été utilisé pour prendre en charge le transport de corps de fragment XML dans la RFC 4825, le protocole d'accès à la configuration XML (XCAP).

Informations supplémentaires :
Numéro magique : aucun
Extension de fichier : .xav
Code de type de fichier Macintosh : "TEXT"
Adresse personnelle et de messagerie pour plus d'informations : Jonathan Rosenberg, jdrosen@jdrosen.net
Utilisation prévue : COURANTE
Auteur/contrôleur des changements : l'IETF.

15.2.3 Type MIME application/xcap-ns+xml

Nom de type de support MIME : application
Nom de sous type MIME : xcap-ns+xml
Paramètres obligatoires : aucun
Paramètres facultatifs : les mêmes que le paramètre de jeu de caractère application/xml comme spécifié dans la [RFC3023].
Considérations de codage : les mêmes que les considérations de codage de application/xml comme spécifié dans la [RFC3023].
Considérations de sécurité : voir la Section 10 de la [RFC3023].
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 4825
Applications qui utilisent ce type de supports : le présent type de document a été utilisé pour prendre en charge le transport de corps de fragment XML dans la RFC 4825, le protocole d'accès à la configuration XML (XCAP).

Informations supplémentaires :
Numéro magique : aucun
Extension de fichier : .xns
Code de type de fichier Macintosh : "TEXT"
Adresse personnelle et de messagerie pour plus d'informations : Jonathan Rosenberg, jdrosen@jdrosen.net
Utilisation prévue : COURANTE
Auteur/contrôleur des changements : l'IETF.

15.2.4 Type MIME application/xcap-error+xml

Nom de type de support MIME : application
Nom de sous type MIME : xcap-error+xml
Paramètres obligatoires : aucun
Paramètres facultatifs : les mêmes que le paramètre de jeu de caractère application/xml comme spécifié dans la [RFC3023].
Considérations de codage : les mêmes que les considérations de codage de application/xml comme spécifié dans la [RFC3023].
Considérations de sécurité : voir la Section 10 de la [RFC3023].
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 4825
Applications qui utilisent ce type de supports : le présent type de document porte les conditions d'erreur définies dans la RFC 4825

Informations supplémentaires :
Numéro magique : aucun
Extension de fichier : .xer
Code de type de fichier Macintosh : "TEXT"
Adresse personnelle et de messagerie pour plus d'informations : Jonathan Rosenberg, jdrosen@jdrosen.net
Utilisation prévue : COURANTE
Auteur/contrôleur des changements : l'IETF.

15.2.5 Type MIME application/xcap-caps+xml

Nom de type de support MIME : application
Nom de sous type MIME : xcap-caps+xml
Paramètres obligatoires : aucun
Paramètres facultatifs : les mêmes que le paramètre de jeu de caractère application/xml comme spécifié dans la [RFC3023].
Considérations de codage : les mêmes que les considérations de codage de application/xml comme spécifié dans la [RFC3023].
Considérations de sécurité : voir la Section 10 de la [RFC3023].
Considérations d'interopérabilité : aucune
Spécification publiée : RFC 4825
Applications qui utilisent ce type de supports : Le présent type de document porte les capacités d'un serveur du protocole d'accès à la configuration XML (XCAP) comme définies dans la RFC 4825.

Informations supplémentaires :
Numéro magique : aucun

Extension de fichier : .xca
Code de type de fichier Macintosh : "TEXT"
Adresse personnelle et de messagerie pour plus d'informations : Jonathan Rosenberg, jdrosen@jdrosen.net
Utilisation prévue : COURANTE
Auteur/contrôleur des changements : l'IETF.

15.3 Enregistrements de sous espace de noms d'URN

La présente spécification enregistre plusieurs nouveaux espaces de noms XML, selon les lignes directrices de la [RFC3688].

15.3.1 urn:ietf:params:xml:ns:xcap-error

URI : l'URI pour cet espace de noms est urn:ietf:params:xml:ns:xcap-error
Contact d'enregistrement : IETF, groupe de travail SIMPLE, (simple@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).

XML :

DÉBUT

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <meta http-equiv="content-type"
   content="text/html;charset=iso-8859-1"/>
 <title>Espace de noms d'erreur XCAP</title>
</head>
<body>
 <h1>Espace de noms pour les documents d'erreur XCAP</h1>
 <h2>urn:ietf:params:xml:ns:xcap-error</h2>
 <p>Voir <a href="http://www.rfc-editor.org/rfc/rfc4825.txt">
   RFC4825</a></p>
</body>
</html>
```

FIN

15.3.2 urn:ietf:params:xml:ns:xcap-caps

URI : l'URI pour cet espace de noms est urn:ietf:params:xml:ns:xcap-caps
Contact d'enregistrement : IETF, groupe de travail SIMPLE, (simple@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).

XML :

DÉBUT

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
  "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <meta http-equiv="content-type"
   content="text/html;charset=iso-8859-1"/>
 <title>Espace de noms de capacités XCAP</title>
</head>
<body>
 <h1>Espace de noms pour les documents de capacités XCAP</h1>
 <h2>urn:ietf:params:xml:ns:xcap-caps</h2>
 <p>See <a href="http://www.rfc-editor.org/rfc/rfc4825.txt">
   RFC4825</a></p>
```

```
</body>
</html>
FIN
```

15.4 Enregistrements de schéma XML

Cette section enregistre deux schémas XML selon les procédures de la [RFC3688].

15.4.1 Enregistrement de schéma d'erreur XCAP

URI : urn:ietf:params:xml:schema:xcap-error
Contact d'enregistrement : IETF, groupe de travail SIMPLE, (simple@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).
Schéma XML : le XML pour ce schéma peut être trouvé comme seul contenu du paragraphe 11.2.

15.4.2 Enregistrement de schéma de capacités XCAP

URI : urn:ietf:params:xml:schema:xcap-caps
Contact d'enregistrement : IETF, groupe de travail SIMPLE, (simple@ietf.org), Jonathan Rosenberg (jdrosen@jdrosen.net).
Schéma XML : le XML pour ce schéma peut être trouvé comme seul contenu du paragraphe 12.2.

16. Remerciements

L'auteur tient à remercier Jari Urpalainen, qui a contribué par beaucoup d'importants commentaires et a aidé à l'édition de ce document. L'auteur remercie aussi Ben Campbell, Eva-Maria Leppanen, Hisham Khartabil, Chris Newman, Joel Halpern, Lisa Dusseault, Tim Bray, Pete Cordell, Jeroen van Bemmel, Christian Schmidt, et Spencer Dawkins de leurs apports et commentaires. Un merci tout spécial à Ted Hardie pour ses apports et son soutien.

17. Références

17.1 Références normatives

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

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

[RFC2617] J. Franks et autres, "Authentification HTTP : Authentification d'accès de base et par résumé", juin 1999. *(DS.)*

[RFC2817] R. Khare, S. Lawrence, "Mise à niveau de TLS au sein de HTTP/1.1", mai 2000. *(P.S.)*

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

[RFC3023] M. Murata, S. St.Laurent et D. Kohn, "Types de support XML", janvier 2001. (*Obsolète, voir* RFC7303)

[RFC3261] J. Rosenberg et autres, "SIP : Protocole d'initialisation de session", juin 2002. (*Mise à jour par* 3265, 3853, 4320, 4916, 5393, 6665, 8217, 8760)

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

[RFC3688] M. Mealling, "Registre XML de l'IETF", BCP 81, janvier 2004.

[RFC3986] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiant de ressource uniforme (URI) : Syntaxe générique", STD 66, janvier 2005. *(P.S. ; MàJ par* RFC8820)

[RFC4234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", octobre 2005. *(Remplace* RFC2234, *remplacée par* RFC5234)

[RFC4288] N. Freed et J. Klensin, "Spécifications du type de support et procédures d'enregistrement", BCP 13, décembre 2005.

[XML] Maler, E., Yergeau, F., Paoli, J., Bray, T., and C. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0 (Third Edition)", World Wide Web Consortium FirstEdition REC-xml-20040204, février 2004, <http://www.w3.org/TR/2004/REC-xml-20040204 >.

[XML-Canon] Boyer, J., "Canonical XML Version 1.0", World Wide Web Consortium Recommendation REC-xml-c14n-20010315, mars 2001, <http://www.w3.org/TR/2001/REC-xml-c14n-20010315 >.

[XMLns] Daniel, R., DeRose, S., Maler, E., and J. Marsh, "XPointer xmlns() Scheme", World Wide Web Consortium Recommendation REC-xptr-xmlns-20030325, mars 2003, <http://www.w3.org/TR/2003/REC-xptr-xmlns-20030325 >.

[XML-Noms] Layman, A., Hollander, D., and T. Bray, "Namespaces in XML", World Wide Web Consortium FirstEdition REC-xml-names-19990114, janvier 1999, <http://www.w3.org/TR/1999/REC-xml-names-19990114 >.

[XML-Struct] Thompson, H., Maloney, M., Mendelsohn, N., and D. Beech, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, octobre 2004, <http://www.w3.org/TR/2004/REC-xmlschema-1-20041028 >.

[XPath] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, novembre 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

[Xpointer] Grosso, P., Marsh, J., Maler, E., and N. Walsh, "XPointer Framework", World Wide Web Consortium Recommendation REC-xptr-framework-20030325, mars 2003, <http://www.w3.org/TR/2003/REC-xptr-framework-20030325 >.

17.2 Références pour information

[RFC2244] C. Newman, J. G. Myers,"ACAP – Protocole d'accès à la configuration d'application", novembre 1997. *(P.S.)*

[RFC2434] T. Narten et H. Alvestrand, "Lignes directrices pour la rédaction d'une section Considérations relatives à l'IANA dans les RFC", BCP 26, octobre 1998. *(Rendue obsolète par la* RFC5226)

[RFC2778] M. Day, J. Rosenberg et H. Sugano, "Modèle pour Presence et la messagerie instantanée", février 2000.

[RFC3265] A.B. Roach, "Notification d'événement spécifique du protocole d'initialisation de session (SIP)", juin 2002. (*MàJ par* RFC6446) (*Remplacée par la* RFC6665)

[RFC3856] J. Rosenberg, "Paquetage d'événement Presence pour le protocole d'initialisation de session (SIP)", août 2004.

[RFC3987] M. Duerst et M. Suignard, "Identifiant de ressource internationalisé (IRI)", janvier 2005.

[RFC4662] A. B. Roach et autres, "Extension de notification d'événement du protocole d'initialisation de session (SIP) pour les listes de ressources", août 2006. *(P.S.)*

[RFC4826] J. Rosenberg, "Formats du langage de balisage extensible (XML) pour représenter des listes de ressources", mai 2007. *(P.S.)*

[XML-Frag] Grosso, P. and D. Veillard, "XML Fragment Interchange", World Wide Web Consortium CR CR-xml-fragment-20010212, février 2001, <http://www.w3.org/TR/2001/CR-xml-fragment-20010212 >.

[XML-Lang] Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., and J. Simeon, "XML Path Language (XPath) 2.0", World Wide Web Consortium CR http://www.w3.org/TR/2005/CR-xpath20-

20051103 , novembre 2005.

Adresse de l'auteur

Jonathan Rosenberg
Cisco
Edison, NJ
US
mél : jdrosen@cisco.com
URI : http://www.jdrosen.net

Déclaration complète de droits de reproduction

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