Groupe de travail Réseau
Request for Comments : 5730
STD : 69
RFC rendue obsolète : 4930
Catégorie :Norme

S. Hollenbeck, VeriSign, Inc.
août 2009

Traduction Claude Brière de L'Isle

Protocole d'approvisionnement extensible (EPP)

Résumé
Le présent document décrit un protocole client-serveur de couche d'application pour le provisionnement et la gestion des objets mémorisés dans un dépôt central partagé. Spécifié en XML, le protocole définit des opérations génériques de gestion d'objet et un cadre extensible qui transpose les opérations du protocole en objets. Le présent document inclut la spécification du protocole, un gabarit de transposition d'objet, et un enregistrement de type de support XML. Le présent document rend obsolète la RFC 4930.

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

Table des matières

1. Introduction

Le présent document décrit les spécifications pour le protocole d'approvisionnement extensible (EPP, *Extensible Provisioning Protocol*) version 1.0, un protocole de texte XML qui permet à plusieurs fournisseurs de services d'effectuer des opérations d'approvisionnement d'objets en utilisant un répertoire central d'objets partagé. EPP est spécifié en utilisant le langage de balisage extensible (XML, *Extensible Markup Language*) 1.0 tel que décrit dans [W3C.REC-xml-20040204] et la notation de schéma XML telle que décrite dans [W3C.REC-xmlschema-1-20041028] et [W3C.REC-xmlschema-2-20041028]. EPP satisfait et dépasse les exigences pour un protocole générique de registre registraire telles que décrites dans la [RFC3375]. Le présent document rend obsolète la [RFC4930].

Le contenu d'EPP est identifié par le type de support MIME application/epp+xml. Les informations d'enregistrement de ce type de support sont incluses dans un appendice au présent document.

EPP est destiné à être utilisé dans divers environnements de fonctionnement où les exigences de transport et de sécurité varient largement. Il est peu probable qu'une seule spécification de transport ou de sécurité satisfasse aux besoins de tous les opérateurs envisageables, de sorte que EPP a été conçu pour une utilisation dans un environnement de protocole en couches. Les liens avec des protocoles spécifiques de transport et de sécurité sortent du domaine d'application de la présente spécification.

Le motif original de ce protocole était de fournir un protocole standard d'enregistrement des noms de domaine de l'Internet à utiliser entre les registraires de noms de domaine et les registres de noms de domaine. Ce protocole donne un moyen d'interaction entre les applications d'un registraire et les applications de registre. On s'attend à ce que ce protocole ait des utilisations supplémentaires au delà de l'enregistrement des noms de domaine.

XML est sensible à la casse. Sauf mention contraire, les spécifications et exemples XML fournis dans le présent document DOIVENT être interprétés dans la casse de caractères présentée pour développer une mise en œuvre conforme.

1.1 Conventions de notation

Les mots clés "DOIT", "NE DOIT PAS", "EXIGÉ", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDÉ", "PEUT", et "FACULTATIF" dans ce document sont à interpréter comme décrit dans la [RFC2119].

Dans les exemples, "C:" représente les lignes envoyées par un client de protocole et "S:" représente les lignes retournées par un serveur du protocole. Les sauts à la ligne et les espaces ne sont fournis dans les exemples que pour illustrer les relations entre les éléments et ne sont pas des caractéristiques EXIGÉES de ce protocole. Un client de protocole qui est autorisé à gérer un objet existant est décrit comme un client de "parrainage" dans le présent document.

2. Description du protocole

EPP est un protocole XML à états pleins qui peut être mis en couches sur plusieurs protocoles de transport. Protégés par des protocoles de sécurité de couche inférieure, les clients échangent des informations d'identification, d'authentification, et d'options, et s'engagent ensuite dans une série d'échanges de commandes/réponses à l'initiative du client. Toutes les commandes EPP sont atomiques (il n'y a pas de réussite ou échec partiel) et elles sont conçues de telle sorte qu'elle peuvent être rendues idempotentes (exécuter une commande plus d'une fois a le même effet net sur l'état du système que d'exécuter une fois la commande avec succès).

EPP fournit quatre éléments de service de base : la découverte de service, les commandes, les réponses, et un cadre d'extension qui prend en charge la définition des objets gérés et les relations des demandes et réponses du protocole à ces objets.

Un serveur EPP DOIT répondre aux communications initiées par le client (qui peuvent être soit une demande de connexion de couche inférieure, soit un message EPP de découverte de service) en retournant un salut au client. Un serveur DOIT répondre promptement à chaque commande EPP avec une réponse coordonnée qui décrit le résultat du traitement de la commande. Le diagramme d'automate à états de serveur illustre en détails le processus d'échange de messages :

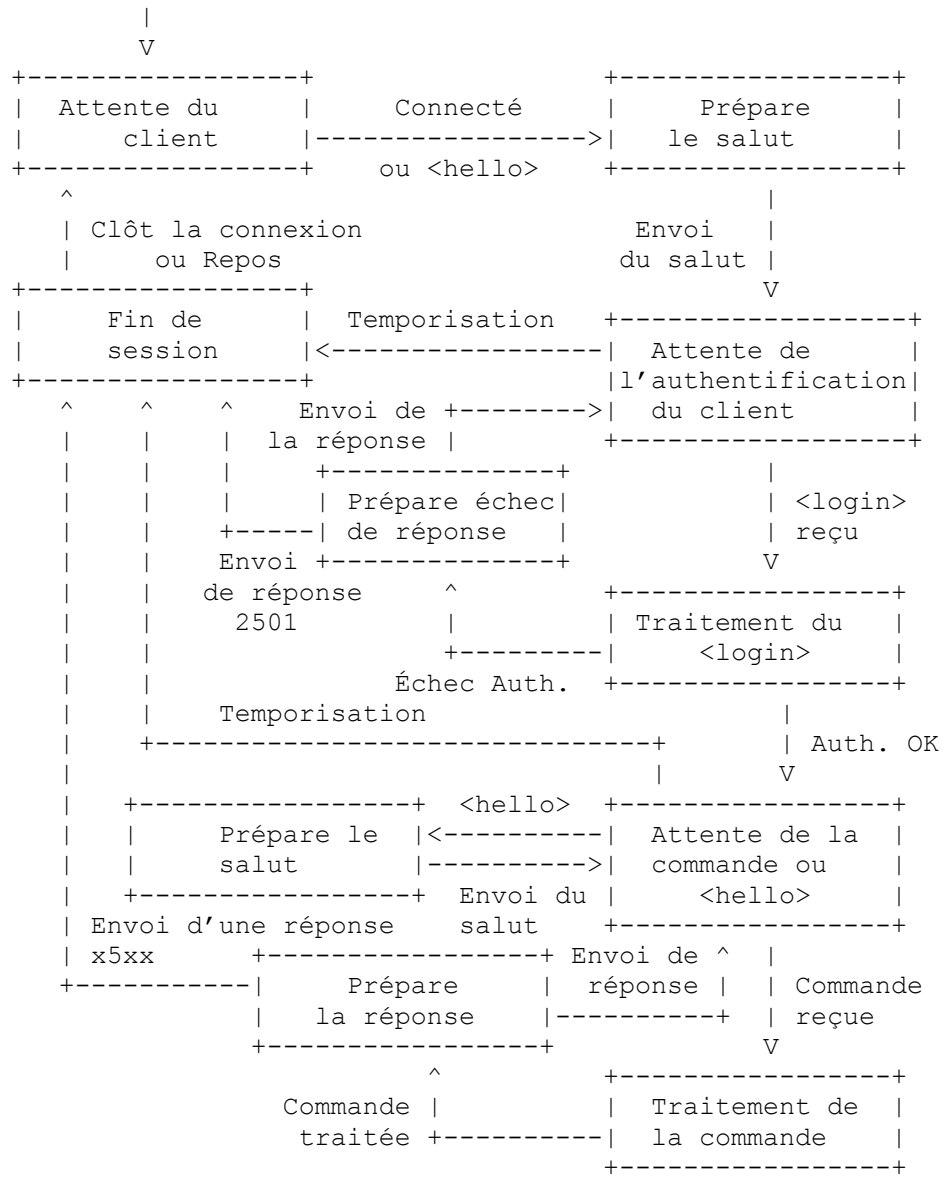

Figure 1 : Automate à états de serveur EPP

Les commandes EPP entrent dans trois catégories : commandes de gestion de session, commandes d'interrogation, et commande de transformation d'objet. Les commandes de gestion de session sont utilisées pour établir et terminer des sessions persistantes avec un serveur EPP. Les commandes d'interrogation sont utilisées pour effectuer des opérations de restitution d'informations en lecture seule sur des objets. Les commandes de transformation sont utilisées pour effectuer des opérations de gestion d'objet en lecture écriture.

Les commandes sont traitées par un serveur dans l'ordre de leur réception d'un client. Bien qu'une réponse immédiate confirmant la réception et le traitement de la commande soit produite par le serveur, le protocole comporte des dispositions pour permettre une révision hors ligne des commandes de transformation avant que l'action demandée soit réellement achevée. Dans de telles situations, la réponse du serveur DOIT clairement noter que la commande a été reçue et traitée mais que l'action demandée est en cours. L'état de l'objet correspondant DOIT clairement refléter le traitement de l'action en cours. Le serveur DOIT aussi notifier au client quand le traitement hors ligne de l'action a été achevé. Les transpositions d'objets DEVRAIENT décrire les formats standard pour des notifications qui décrivent l'achèvement du traitement hors ligne.

EPP utilise les espaces de noms XML pour fournir un cadre extensible de gestion d'objet et pour identifier les schémas requis pour l'analyse et la validation des instances XML. Ces espaces de noms et définitions de schéma sont utilisés pour identifier le schéma de protocole de base et les schémas pour les objets gérés. Les préfixes d'espace de noms XML utilisés dans les exemples (comme la chaîne "foo" dans "xmlns:foo") ne sont destinés qu'à des fins d'illustration. Une mise en œuvre conforme NE DOIT PAS en exiger l'utilisation ni d'aucun autre préfixe d'espace de nom spécifique.

Toutes les instances XML DEVRAIENT commencer par une déclaration <?xml?> pour identifier la version de XML qui est

utilisée, facultativement identifier l'utilisation du codage de caractères utilisé, et facultativement fournir une indication à un analyseur XML qu'un fichier externe de schéma est nécessaire pour valider l'instance XML. Les analyseurs XML conformes reconnaissent l'UTF-8 (défini dans la [RFC3629]) et l'UTF-16 (défini dans la [RFC2781]) ; selon la [RFC2277], l'UTF-8 est le codage de caractères RECOMMANDÉ à utiliser avec EPP.

Les codages de caractères autres que UTF-8 et UTF-16 sont permis par XML. UTF-8 est le codage par défaut supposé par XML en l'absence d'un attribut "codage" ou une marque d'ordre des octets (BOM, *byte order mark*) ; donc, l'attribut "codage" dans la déclaration XML est FACULTATIVE si le codage UTF-8 est utilisé. Les clients et serveurs EPP DOIVENT accepter un BOM UTF-8 s'il est présent, bien que l'émission d'un BOM UTF-8 ne soit PAS RECOMMANDÉE.

Exemples de déclarations XML :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml version="1.0" standalone="no"?>
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0"?>
```

2.1 Considérations sur la transposition de transport

Comme décrit précédemment, EPP peut être mis en couches sur plusieurs protocoles de transport. Il y a cependant un ensemble commun de considérations qui DOIVENT être prises en compte par toute transposition de transport définie pour EPP. Cela inclut :

- La transposition de transport DOIT préserver l'ordre des commandes.
- La transposition de transport DOIT s'adresser à la relation entre les sessions et le concept de connexion client-serveur.
- La transposition de transport DOIT préserver la nature à état pleins du protocole.
- La transposition de transport DOIT tramer les unités de données.
- La transposition de transport DOIT être sur un transport, comme TCP [RFC0793] ou le protocole de transmission de contrôle de commande (SCTP) [RFC4960], assurant l'évitement d'encombrement qui suive la [RFC2914]; ou, si il se transpose sur un protocole tel que SMTP [RFC5321] ou le protocole d'échange extensible de blocs (BEEP) [RFC3080], les questions de performance doivent alors prendre en compte les questions de surcharge, de disponibilité de serveur, et ainsi de suite.
- La transposition de transport DOIT assurer la fiabilité.
- La transposition de transport DOIT permettre ou interdire explicitement le traitement en parallèle.

Le traitement en parallèle *(pipelining)* aussi appelé flux de commandes en continu *(command streaming)* se fait lorsque un client envoie plusieurs commandes à un serveur sans attendre la réponse correspondant à chacune. Après l'envoi des commandes, le client attend que les réponses arrivent dans l'ordre correspondant aux commandes achevées. Des gains de performance peuvent parfois être réalisés avec le traitement en parallèle, en particulier avec des transports à forte latence, mais il y a des considérations supplémentaires qui sont associées à la définition d'une transposition de transport qui prend en charge le traitement en parallèle :

- les commandes DOIVENT être traitées indépendamment l'une de l'autre,
- selon le transport, le traitement en parallèle PEUT se faire sous la forme de l'envoi d'une session complète dans un "lot" bien défini,
- la transposition de transport DOIT décrire comment une erreur de traitement d'une commande affecte la poursuite du fonctionnement de la session.

Une transposition de transport DOIT expliquer comment toutes ces exigences sont satisfaites, selon le protocole de transport utilisé pour échanger les données.

2.2 Identification du protocole

Toutes les instances XML EPP DOIVENT commencer par un élément <epp>. Cet élément identifie le début d'un élément de transport EPP et l'espace de noms utilisé dans le protocole. L'élément de début <epp> et l'élément </epp> associé de fin DOIVENT être appliqués à toutes les structures envoyées aux clients et aux serveurs.

Exemple d'éléments EPP de "début" et de "fin" :

```
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
</epp>
```

2.3 Format de Hello

EPP PEUT être porté sur des protocoles de transport en mode aussi bien connexion que sans connexion. Un client EPP PEUT demander un <greeting> *(salut)* d'un serveur EPP à tout moment entre une commande <login> réussie et une commande <logout> par l'envoi d'un <hello> à un serveur. L'utilisation de cet élément est essentielle dans un environnement sans connexion où un serveur ne peut pas retourner un <greeting> en réponse à une connexion initiée par un client. Un <hello> EPP DOIT être un élément vide sans élément fils.

Exemple de <hello> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <hello/>
C:</epp>
```

2.4 Format d'accueil

Un serveur EPP répond à une connexion réussie et un élément <hello> en retournant un élément <greeting> au client. Un <greeting> EPP contient les éléments suivants :

- Un élément <svID> qui contient le nom du serveur.
- Un élément <svDate> qui contient la date et l'heure en cours du serveur en temps universel coordonné (UTC).
- Un élément <svcMenu> qui identifie les services pris en charge par le serveur, incluant :
  - un ou plusieurs éléments <version> qui identifient les versions de protocole supportées par le serveur,
  - un ou plusieurs éléments <lang> qui contiennent les identifiants des langages de réponse de texte connus par le serveur. Les identifiants de langage DOIVENT être structurés comme documenté dans la [RFC4646],
  - un ou plusieurs éléments <objURI> qui contiennent les URI d'espace de noms représentant les objets que le serveur est capable de gérer. Un serveur PEUT limiter les privilèges de gestion d'objet client par client.
  - un élément FACULTATIF <svcExtension> qui contient un ou plusieurs éléments <extURI> qui contiennent des URI d'espace de noms représentant des extensions d'objet supportées par le serveur,
  - un élément <dcp> (*data collection policy*, politique de collection de données) qui contient des éléments fils utilisés pour décrire la politique de confidentialité du serveur pour la collecte et la gestion des données. Les implications de politique s'étendent habituellement au delà des relations client serveur. Clients et serveurs peuvent tous deux avoir des relations avec d'autres entités qui ont besoin de connaître la politique de collecte de données de l'opérateur du serveur pour prendre des décisions d'approvisionnement en connaissance de cause. Les informations de politique DOIVENT être divulguées aux entités d'approvisionnement, bien que la méthode de divulgation des données de politique en dehors d'une interaction directe de protocole sorte du domaine d'application de la présente spécification. Les éléments fils incluent les suivants :
    * Un élément <access> qui décrit l'accès fourni par le serveur au client au nom de la source d'origine des données. L'élément <access> DOIT contenir un des éléments fils suivants :
      + <all/> : l'accès est donné à toutes les données identifiées.
      + <none/> : aucun accès n'est fourni aux données identifiées.
      + <null/> : les données ne sont pas persistantes, de sorte qu'aucun accès n'est possible.
      + <personal/> : l'accès est donné aux données identifiées relatives aux entités individuelles et organisationnelles.
      + <personalAndOther/> : l'accès est donné aux données identifiées relatives aux entités individuelles, organisationnelles, et autres données de nature non personnelle.
      + <other/> : l'accès est donné aux autres données identifiées de nature non personnelle.
    * Un ou plusieurs éléments <statement> qui décrivent l'objet de la collecte des données, les receveurs des données, et la conservation des données. Chaque élément <statement> DOIT contenir un élément <purpose>, un élément <recipient>, et un élément <retention>. L'élément <purpose> DOIT contenir un ou plusieurs des éléments fils suivants qui décrivent l'objet pour lequel les données sont collectées :
      + <admin/> : objet administratif. Les informations peuvent être utilisées pour le soutien administratif et technique du système d'approvisionnement.
      + <contact/> : contact pour des besoins commerciaux. Les informations peuvent être utilisées pour contacter des individus, par un canal de communications autre que le protocole, pour la promotion d'un produit ou service.
      + <prov/> : à des fins d'approvisionnement d'objet. Les informations peuvent être utilisées pour identifier des objets et des relations entre les objets.
      + <other/> : à d'autres fins. Les informations peuvent être utilisées à d'autres fins non capturées par les définitions précédentes.
    * L'élément <recipient> DOIT contenir un ou plusieurs des éléments fils suivants qui décrivent les receveurs des données collectées :
      + <other/> : d'autres entités suivant des pratiques inconnues.
      + <ours> : l'opérateur du serveur et/ou des entités agissant comme agents ou entités pour lesquelles l'opérateur du

serveur agit comme agent. Un agent dans cette instance est défini comme un tiers qui ne traite les données qu'au nom du fournisseur de service pour l'exécution des fins déclarées. L'élément <ours> contient un élément FACULTATIF <recDesc> qui peut être utilisé pour décrire le receveur.
    + <public/> : des forums publics.
    + <same/> : d'autres entités qui suivent les pratiques du serveur.
    + <unrelated/> : des tiers sans relations.
  * L'élément <retention> DOIT contenir un des éléments fils suivants qui décrivent les pratiques de conservation des données :
    + <business/> : Les données persistent selon les pratiques d'affaires.
    + <indefinite/> : Les données persistent indéfiniment.
    + <legal/> : Les données persistent selon les exigences légales.
    + <none/> : Les données ne sont pas persistantes et ne sont pas conservées plus que la brève période nécessaire pour les utiliser durant le cours d'une seule interaction en ligne.
    + <stated/> : Les données persistent pour satisfaire l'objet déclaré.
  * Un élément FACULTATIF <expiry> qui décrit la durée de vie de la politique. L'élément <expiry> DOIT contenir un des éléments fils suivants :
    + <absolute/> : La politique est valide depuis la date et heure courante jusqu'à ce qu'elle arrive à expiration à la date et heure spécifiées.
    + <relative/> : La politique est valide depuis la date et heure courante jusqu'à la fin de la durée spécifiée.

Les éléments de politique de collecte de données se fondent sur les travaux décrits dans la plate-forme de spécification des préférences de confidentialité du World Wide Web Consortium [W3C.REC-P3P-20020416].

Exemple de salut :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <greeting>
S:    <svID>Exemple de serveur EPP epp.example.com</svID>
S:    <svDate>2000-06-08T22:00:00.0Z</svDate>
S:    <svcMenu>
S:      <version>1.0</version>
S:      <lang>en</lang>
S:      <lang>fr</lang>
S:      <objURI>urn:ietf:params:xml:ns:obj1</objURI>
S:      <objURI>urn:ietf:params:xml:ns:obj2</objURI>
S:      <objURI>urn:ietf:params:xml:ns:obj3</objURI>
S:      <svcExtension>
S:        <extURI>http://custom/obj1ext-1.0</extURI>
S:      </svcExtension>
S:    </svcMenu>
S:    <dcp>
S:      <access><all/></access>
S:      <statement>
S:        <purpose><admin/><prov/></purpose>
S:        <recipient><ours/><public/></recipient>
S:        <retention><stated/></retention>
S:      </statement>
S:    </dcp>
S:  </greeting>
S:</epp>
```

2.5 Format de commande

Un client EPP interagit avec un serveur EPP en envoyant une commande au serveur et en recevant une réponse du serveur. En plus des éléments EPP standard, une commande EPP contient les éléments suivants :

- Un élément de commande dont l'étiquette correspond à une des commandes EPP valides décrites dans le présent document. L'élément de commande PEUT contenir des éléments fils spécifiés soit par le protocole, soit par un objet.
- Un élément FACULTATIF <extension> qui PEUT être utilisé pour des extensions de commande définies par le serveur.
- Un élément FACULTATIF <clTRID> (identifiant de transaction de client) qui PEUT être utilisé pour identifier de façon univoque la commande au client. Les clients sont responsables de la conservation de leur propre espace d'identifiant de transaction pour assurer l'unicité.

Exemple de commande avec des éléments fils spécifiés par l'objet :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <obj:info xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <obj:name>example</obj:name>
C:      </obj:info>
C:    </info>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

2.6 Format de réponse

Un serveur EPP répond à une commande de client en retournant une réponse au client. Les commandes EPP sont atomiques, de sorte qu'une commande va soit réussir complètement, soit échouer complètement. Les résultats de succès et d'échec NE DOIVENT PAS être mélangés. En plus des éléments EPP standard, une réponse EPP contient les éléments suivants :

- Un ou plusieurs éléments <result> qui documentent le succès ou l'échec de l'exécution de la commande. Si la commande a été traitée avec succès, un seul élément <result> DOIT être retourné. Si la commande n'a pas été traitée avec succès, plusieurs éléments <result> PEUVENT être retournés pour documenter les conditions d'échec. Chaque élément <result> contient les attributs et éléments fils suivants :
  - Un attribut "code" dont la valeur est un nombre décimal à quatre chiffres qui décrit le succès ou l'échec de la commande.
  - Un élément <msg> qui contient une description lisible par l'homme du code de réponse. Le langage de la réponse est identifié via un attribut FACULTATIF "lang". Si il n'est pas spécifié, la valeur d'attribut par défaut DOIT être "en" (anglais).
  - Zéro, un ou plusieurs éléments FACULTATIFS <value> qui identifient un élément fourni par le client (incluant l'étiquette et la valeur XML) ou d'autres informations sur ce qui a causé la condition d'erreur du serveur.
  - Zéro, un ou plusieurs éléments FACULTATIFS <extValue> qui peuvent être utilisés pour fournir des informations supplémentaires de diagnostic d'erreur, incluant :
    * Un élément <value> qui identifie un élément fourni par le client (incluant l'étiquette et la valeur XML) qui a causé la condition d'erreur du serveur.
    * Un élément <reason> qui contient un message lisible par l'homme qui décrit la raison de l'erreur. Le langage de la réponse est identifié via un attribut FACULTATIF "lang". Si il n'est pas spécifié, la valeur par défaut de l'attribut DOIT être "en" (anglais).

- Un élément FACULTATIF <msgQ> qui décrit les messages mis en file d'attente pour être restitués au client. Un élément <msgQ> NE DOIT PAS être présent si il n'y a pas de message en file d'attente pour être restitué au client. Un élément <msgQ> PEUT être présent dans les réponses aux commandes EPP autres que la commande <poll> si des messages sont en file d'attente pour restitution. Un élément <msgQ> DOIT être présent dans les réponses à la commande EPP <poll> si les messages sont en file d'attente pour restitution. L'élément <msgQ> contient les attributs suivants :
  - Un attribut "count" qui décrit le nombre de messages qui existent dans la file d'attente.
  - Un attribut "id" utilisé pour identifier de façon univoque le message en tête de la file d'attente.

  L'élément <msgQ> contient les éléments fils FACULTATIFS suivants qui DOIVENT être retournés en réponse à une commande de demande <poll> et NE DOIVENT PAS être retournés en réponse à toute autre commande, incluant un accusé de réception <poll> :
  - Un élément <qDate> qui contient la date et l'heure à laquelle le message a été mis en file d'attente.
  - Un élément <msg> contenant un message lisible par l'homme. Le langage de la réponse est identifié via un attribut FACULTATIF "lang". Si il n'est pas spécifié, la valeur de l'attribut par défaut DOIT être "en" (anglais). Cet élément PEUT contenir un contenu XML pour des besoins de formatage, mais le contenu XML n'est pas spécifié par le protocole et sa validité ne sera donc pas vérifiée.

- Un élément FACULTATIF <resData> (données de réponse) qui contient des éléments fils spécifiques de la commande et de l'objet associé.

- Un élément FACULTATIF <extension> qui PEUT être utilisé pour des extensions de réponse définies par le serveur.

- Un élément <trID> (identifiant de transaction) contenant l'identifiant de transaction alloué par le serveur à la commande pour laquelle la réponse est retournée. L'identifiant de transaction est formé en utilisant le <clTRID> associé à la commande si elle est fournie par le client et un <svTRID> (identifiant de transaction de serveur) qui est alloué de façon

univoque par le serveur.

Les identifiants de transaction fournissent le moyen de vérifier l'intégrité de la synchronisation commande-réponse. Ils DEVRAIENT être enregistrés, conservés et protégés pour assurer que le client et le serveur ont tous deux des enregistrements cohérents de la gestion d'état et du temps.

Exemple de réponse sans <value> ou <resData> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg lang="fr">Commande achevée avec succès</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Exemple de réponse avec <resData> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <resData>
S:      <obj:creData xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <obj:name>example</obj:name>
S:      </obj:creData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Exemple de réponse avec des éléments de valeur d'erreur :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="2004">
S:      <msg>Erreur de gamme de la valeur de paramètre</msg>
S:      <value xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <obj:elem1>2525</obj:elem1>
S:      </value>
S:    </result>
S:    <result code="2005">
S:      <msg>Erreur de syntaxe de la valeur de paramètre</msg>
S:      <value xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <obj:elem2>example</obj:elem2>
S:      </value>
S:      <extValue>
S:        <value xmlns:obj="urn:ietf:params:xml:ns:obj">
S:          <obj:elem3>abc.ex(ample</obj:elem3>
S:        </value>
S:        <reason>Trouvé un caractère invalide.</reason>
S:      </extValue>
S:    </result>
S:    <trID>
```

```
S:     <clTRID>ABC-12345</clTRID>
S:     <svTRID>54321-XYZ</svTRID>
S:   </trID>
S:  </response>
S:</epp>
```

Exemple de réponse avec notice de messages de serveur en attente :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:   <result code="1000">
S:     <msg>Commande achevée avec succès</msg>
S:   </result>
S:   <msgQ count="5" id="12345"/>
S:   <trID>
S:     <clTRID>ABC-12345</clTRID>
S:     <svTRID>54321-XYZ</svTRID>
S:   </trID>
S:  </response>
S:</epp>
```

Le succès ou l'échec d'une commande NE DOIT pas être supposé si aucune réponse n'est retournée ou si une réponse retournée est mal formée. L'idempotence du protocole assure la sécurité du réessai d'une commande dans les cas d'échec de livraison de la réponse.

2.7 Cadre pour l'extension du protocole

EPP fournit un cadre d'extension qui permet d'ajouter des caractéristiques aux niveaux du protocole, de l'objet, et de la commande/réponse.

2.7.1 Extension du protocole

Le cadre d'extension EPP permet la définition de nouveaux éléments du protocole identifiés en utilisant la notation d'espace de noms XML avec une référence à un schéma XML qui définit l'espace de noms. L'élément <epp> qui identifie le début d'une instance de protocole inclut plusieurs choix d'éléments fils, dont un est un élément <extension> dont les enfants définissent l'extension. Par exemple, un élément d'extension de protocole serait décrit en termes génériques comme suit :

```
C:<epp>
C:  <extension>
C:    <!-- Un ou plusieurs éléments d'extension. -->
C:    <ext:foo xmlns:ext="urn:ietf:params:xml:ns:ext">
C:      <!-- Un ou plusieurs élémentts d'extension fils. -->
C:    </ext:foo>
C:  </extension>
C:</epp>
```

Le présent document ne définit pas les transpositions pour des extensions spécifiques. Les spécifications d'extension DOIVENT être décrites dans des documents séparés qui définiront les objets et les opérations sujets de l'extension.

2.7.2 Extension d'objet

EPP fournit un cadre de gestion d'objet extensible qui définit la syntaxe et la sémantique des opérations du protocole appliquées à un objet géré. Ce cadre pousse la définition de chaque opération du protocole dans le contexte d'un objet spécifique, donnant la capacité d'ajouter des transpositions pour les nouveaux objets sans avoir à modifier le protocole de base.

Les éléments de protocole qui contiennent des données spécifiques des objets sont identifiés en utilisant la notation d'espace de noms XML avec une référence à un schéma XML qui définit cet espace de noms. Le schéma pour la prise en charge d'EPP utilise des schémas dynamiques d'objet commande par commande et réponse par réponse. Par exemple, le début d'un élément de commande spécifique d'un objet serait décrit en termes génériques comme suit :

```
C:<EPPCommandName>
```

```
C:  <object:command xmlns:object="urn:ietf:params:xml:ns:object">
C:    <!-- Un ou plusieurs éléments de commande spécifique d'objet. -->
C:  </object:command>
C:</EPPCommandName>
```

Un élément de réponse spécifique d'un objet serait décrit de façon similaire :

```
S:<resData>
S:  <object:resData xmlns:object="urn:ietf:params:xml:ns:object">
S:    <!-- Un ou plusieurs éléments de réponse spécifique d'objet. -->
S:  </object:resData>
S:</resData>
```

Le présent document ne définit pas les transpositions pour des objets spécifiques. La transposition d'EPP en un objet DOIT être décrite dans des documents séparés qui s'adressent spécifiquement à chaque commande et réponse dans le contexte de l'objet. Une présentation suggérée de transposition d'objet est incluse en appendice au présent document.

2.7.3 Extension de commande-réponse

EPP fournit une facilité pour les extensions de commande et réponse du protocole. Les commandes et réponses du protocole PEUVENT être étendues par un élément <extension> qui contient des éléments supplémentaires dont la syntaxe et la sémantique ne sont pas explicitement définies par EPP ou une transposition d'objet EPP. Cet élément est FACULTATIF. Les extensions sont normalement définies par accord entre client et serveur et PEUVENT être utilisées pour étendre EPP pour des besoins de fonctionnement uniques. Un élément de commande étendu par le serveur serait décrit en termes génériques de la façon suivante :

```
C:<command>
C:  <!—Le nom de commande EPP peut être "create", "update", etc. -->
C:  <EPPCommandName>
C:    <object:command xmlns:object="urn:ietf:params:xml:ns:object">
C:      <!-- Un ou plusieurs éléments de commande spécifiques de l'objet. -->
C:    </object:command>
C:  </EPPCommandName>
C:  <extension>
C:    <!-- Un ou plusieurs éléments définis par le serveur. -->
C:  </extension>
C:</command>
```

Un élément de réponse étendu par le serveur serait décrit de façon similaire par :

```
S:<response>
S:  <result code="1000">
S:    <msg lang="fr">Commande achevée avec succès</msg>
S:  </result>
S:  <extension>
S:    <!-- Un ou plusieurs éléments définis par le serveur. -->
S:  </extension>
S:  <trID>
S:    <clTRID>ABC-12345</clTRID>
S:    <svTRID>54321-XYZ</svTRID>
S:  </trID>
S:</response>
```

Le présent document ne définit aucune extension spécifique de serveur. La transposition des extensions de serveur en EPP DOIT être décrite dans des documents séparés qui s'adressent spécifiquement aux commandes et réponses étendues dans le contexte de fonctionnement du serveur.

2.8 Identification d'objet

Certains objets, comme les serveurs de noms et les contacts, peuvent être utiles dans de multiples répertoires. Cependant, conserver des copies disjointes des informations d'objet dans plusieurs répertoires peut conduire à des incohérences qui ont des conséquences néfastes pour l'Internet. Par exemple, changer le nom d'un serveur de noms dans un répertoire mais pas dans un autre qui se réfère au serveur pour la délégation de nom de serveur peut produire des résultats inattendus d'interrogation du DNS.

Des identifiants uniques au monde peuvent faciliter le partage des informations d'objets entre les répertoires. Un identifiant unique au monde DOIT être alloué à chaque objet à sa création ; l'identifiant DOIT être retourné au client au titre de toute demande de restitution des attributs détaillés d'un objet. Les valeurs d'identifiant spécifiques sont l'affaire des politiques de répertoires, mais elles DEVRAIENT être construites selon l'algorithme suivant :

a. Diviser l'univers du répertoire d'approvisionnement en un certain nombre de classes de répertoires d'objets.
b. Chaque répertoire au sein d'une classe reçoit un identifiant qui est tenu par l'IANA.
c. Chaque répertoire est chargé d'allouer un identifiant local univoque à chaque objet qu'il contient.
d. L'identifiant unique au monde est l'enchaînement de l'identifiant local, suivi par un tiret ("-", valeur ASCII 0x002D), suivi par l'identifiant du répertoire.

2.9 Commandes du protocole

EPP fournit des commandes pour gérer les sessions, restituer les informations d'objet, et effectuer les opérations de transformation sur les objets. Toutes les commandes EPP sont atomiques et conçues de telle façon qu'elles puissent être rendues idempotentes, réussissant complètement ou échouant complètement, et produisant des résultats prévisibles en cas d'exécutions répétées. Cette section décrit chaque commande EPP, incluant des exemples de réponses représentatives du serveur.

2.9.1 Commandes de gestion de session

EPP fournit deux commandes pour la gestion de session : <login> pour établir une session avec un serveur et <logout> pour terminer une session avec un serveur. La commande <login> établit une session de serveur courante qui préserve l'identité et les informations d'autorisation du client pendant la durée de la session.

2.9.1.1 Commande EPP <login>

La commande EPP <login> est utilisée pour établir une session avec un serveur EPP en réponse à un salut produit par le serveur. Une commande <login> DOIT être envoyée à un serveur avant toute autre commande EPP pour établir une session en cours. Un opérateur de serveur PEUT limiter le nombre de tentatives de connexions échouées N, $1 \leq N \leq$ infini, après quoi un échec de connexion résulte en la fermeture de la connexion au serveur (si une connexion existe).

Un identifiant de client et un mot de passe initial DOIVENT être créés sur le serveur avant qu'un client puisse réussir à achever une commande <login>. L'identifiant de client et le mot de passe initial DOIVENT être livrés au client en utilisant une méthode hors bande qui protège l'identifiant et le mot de passe d'une divulgation accidentelle.

En plus des éléments de commande EPP standard, la commande <login> contient les éléments fils suivants :

- un élément <clID> qui contient l'identifiant de client alloué au client par le serveur,
- un élément <pw> qui contient le mot de passe en clair du client. La valeur de cet élément est sensible à la casse,
- un élément FACULTATIF <newPW> qui contient un nouveau mot de passe en clair à allouer au client pour qu'il l'utilise dans les commandes <login> suivantes. La valeur de cet élément est sensible à la casse.
- un élément <options> qui contient les éléments fils suivants :
  - un élément <version> qui contient la version du protocole à utiliser pour la commande ou la session de serveur en cours,
  - un élément <lang> qui contient le langage du texte de réponse à utiliser pour la commande ou les commandes de session de serveur en cours.

  Les valeurs des éléments <version> et <lang> DOIVENT exactement correspondre à une des valeurs présentées dans le salut EPP.
- un élément <svcs> qui contient un ou plusieurs éléments <objURI> qui contiennent les URI d'espace de noms représentant les objets à gérer durant la session. L'élément <svcs> PEUT contenir un élément FACULTATIF <svcExtension> qui contient un ou plusieurs éléments <extURI> qui identifient les extensions d'objet à utiliser durant la session.

Le mécanisme PLAIN de simple authentification et de couche de sécurité (SASL, *Simple Authentication and Security Layer*) présenté dans la [RFC4616] décrit un format pour fournir un identifiant d'usager, un identifiant d'autorisation, et un mot de passe au titre d'une seule chaîne de texte source. Le mécanisme d'authentification EPP est similaire, bien que EPP n'exige pas une identification d'autorisation de niveau session et que l'identifiant et le mot de passe d'utilisateur soient séparés dans des éléments XML distincts. Des schémas d'identification et d'autorisation supplémentaires DOIVENT être fournis à d'autres couches de protocole pour fournir des services de sécurité plus robustes.

Exemple de commande <login> :

C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>

```
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <login>
C:      <clID>ClientX</clID>
C:      <pw>foo-BAR2</pw>
C:      <newPW>bar-FOO2</newPW>
C:      <options>
C:        <version>1.0</version>
C:        <lang>fr</lang>
C:      </options>
C:      <svcs>
C:        <objURI>urn:ietf:params:xml:ns:obj1</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj2</objURI>
C:        <objURI>urn:ietf:params:xml:ns:obj3</objURI>
C:        <svcExtension>
C:          <extURI>http://custom/obj1ext-1.0</extURI>
C:        </svcExtension>
C:      </svcs>
C:    </login>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <login> a été traitée avec succès, la réponse du serveur NE DOIT PAS contenir un élément <resData>. En cas de succès, le serveur va répondre en créant et en entretenant une nouvelle session qui DEVRAIT être terminée par une future commande <logout>.

Exemple de réponse <login> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <login> est utilisée pour établir une session avec un serveur EPP. Une commande <login> DOIT être rejetée si elle est reçue dans les limites d'une session existante. Cette commande DOIT être disponible à tous les clients.

2.9.1.2 Commande EPP <logout>

La commande EPP <logout> est utilisée pour terminer une session avec un serveur EPP. La commande <logout> DOIT être représentée comme un élément vide sans élément fils.

Un serveur PEUT terminer une session à cause de l'inactivité d'un client ou de la longueur excessive de la session d'un client. Les paramètres pour déterminer une inactivité ou une longueur de session excessives d'un client sont une affaire de politique de serveur et ne sont pas spécifiés par le protocole.

Les transpositions de transport DOIVENT décrire explicitement tout traitement de mode connexion qui a lieu après le traitement d'une commande <logout> et qui termine une session.

Exemple de commande <logout> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
```

```
C:   <logout/>
C:   <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <logout> a été traitée avec succès, un serveur DOIT répondre par une réponse EPP sans élément <resData>. Si elle réussit, le serveur DOIT aussi terminer la session en cours.

Exemple de réponse <logout> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1500">
S:      <msg>Commande achevée avec succès ; fin de session</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <logout> est utilisée pour terminer une session avec un serveur EPP. Une commande <logout> DOIT être rejetée si la commande n'a pas été précédée par une commande <login> réussie. Cette commande DOIT être disponible à tous les clients.

2.9.2 Commandes d'interrogation

2.9.2.1 Commande EPP <check>

La commande EPP <check> est utilisée pour déterminer si un objet peut être provisionné au sein d'un répertoire. Elle fournit une indication qui permet à un client d'anticiper le succès ou l'échec de l'approvisionnement d'un objet en utilisant la commande <create> lorsque les exigences d'approvisionnement d'un objet sont en fin de compte une affaire de politique de serveur.

Les éléments nécessaires pour identifier un objet sont spécifiques de l'objet, de sorte que les éléments fils de la commande <check> sont spécifiés en utilisant le cadre d'extension EPP. En plus des éléments standard de commande EPP, la commande <check> contient les éléments fils suivants :

- Un élément <obj:check> spécifique de l'objet qui identifie les objets à rechercher. Plusieurs objets du même type PEUVENT être recherchés au sein d'une seule commande <check>.

Exemple de commande <check> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <check>
C:      <obj:check xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <obj:name>example1</obj:name>
C:        <obj:name>example2</obj:name>
C:        <obj:name>example3</obj:name>
C:      </obj:check>
C:    </check>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <check> a été traitée avec succès, un serveur DOIT répondre avec un élément EPP <resData> qui DOIT contenir un élément fils qui identifie l'espace de nom de l'objet. Les éléments fils de l'élément <resData> sont spécifiques de l'objet, bien qu'un élément EPP <resData> DOIVE contenir un élément <obj:chkData> fils qui contient un ou plusieurs éléments <obj:cd> (check data). Chaque élément <obj:cd> contient les éléments fils suivants :

- Un élément spécifique de l'objet qui identifie l'objet demandé. Cet élément DOIT contenir un attribut "avail" dont la valeur indique la disponibilité de l'objet (peut il être provisionné ou non) au moment de l'achèvement de la commande <check>. Une valeur de "1" ou de "vrai" signifie que l'objet peut être provisionné. Une valeur de "0" ou de "faux" signifie que l'objet ne peut pas être provisionné.
- Un élément FACULTATIF <obj:reason> qui PEUT être fourni lorsque un objet ne peut pas être provisionné. S'il est présent, cet élément contient un texte spécifique du serveur pour aider à expliquer pourquoi l'objet ne peut pas être provisionné. Ce texte DOIT être représenté dans le langage de réponse précédemment négocié avec le client ; un attribut FACULTATIF "lang" PEUT être présent pour identifier le langage si la valeur négociée est quelque chose d'autre que la valeur par défaut de "en" (anglais).

Exemple de réponse <check> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <resData>
S:      <obj:chkData xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <obj:cd>
S:          <obj:name avail="1">example1</obj:name>
S:        </obj:cd>
S:        <obj:cd>
S:          <obj:name avail="0">example2</obj:name>
S:          <obj:reason>In use</obj:reason>
S:        </obj:cd>
S:        <obj:cd>
S:          <obj:name avail="1">example3</obj:name>
S:        </obj:cd>
S:      </obj:chkData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <check> est utilisée pour déterminer si un objet peut être provisionné dans un répertoire. Cette action DOIT être ouverte à tous les clients autorisés.

2.9.2.2 Commande EPP <info>

La commande EPP <info> est utilisée pour restituer les informations associées à un objet existant. Les éléments nécessaires pour identifier un objet et le type d'informations associées à un objet sont spécifiques de l'objet, de sorte que les éléments fils de la commande <info> sont spécifiés en utilisant le cadre d'extension EPP. En plus des éléments standard de commande EPP, la commande <info> contient les éléments fils suivants :

- Un élément <obj:info> spécifique de l'objet qui identifie l'objet recherché.

Exemple de commande <info> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <info>
C:      <obj:info xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <!-- Éléments spécifiques de l'objet. -->
C:      </obj:info>
C:    </info>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <info> a été traitée avec succès, un serveur DOIT répondre avec un élément EPP <resData> qui DOIT contenir un élément fils qui identifie l'espace de noms de l'objet et l'identifiant d'objet de répertoire (ROID, *Repository Object IDentifier*) qui a été alloué à l'objet à sa création. Les autres éléments fils de l'élément <resData> sont spécifiques de l'objet.

Exemple de réponse <info> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <resData>
S:      <obj:infData xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <obj:roid>EXAMPLE1-REP</obj:roid>
S:        <!-- Éléments spécifiques de l'objet. -->
S:      </obj:infData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <info> est utilisée pour restituer les informations associées à un objet existant. Cette action DEVRAIT être limitée aux clients autorisés ; restreindre cette action au client parrain est RECOMMANDÉ.

2.9.2.3 Commande EPP <poll>

La commande EPP <poll> est utilisée pour découvrir et restituer les messages de service mis en file d'attente par un serveur pour les clients individuels. Si la file d'attente de messages n'est pas vide, une réponse de succès à une commande <poll> DOIT retourner le premier message de la file d'attente des messages. Chaque réponse retournée du serveur comporte un identifiant de message univoque pour le serveur qui DOIT être fourni pour accuser réception du message, et un compteur qui indique le nombre de messages dans la file d'attente. Après la réception d'un message par le client, celui-ci DOIT répondre au message par un accusé de réception explicite pour confirmer que le message a été reçu. Un serveur DOIT sortir le message de la file d'attente et décrémenter le compteur de la file d'attente après avoir reçu l'accusé de réception du client, rendant le prochain message de la file d'attente (s'il en est) disponible pour restitution.

Les serveurs peuvent occasionnellement effectuer des actions sur des objets qui ne sont pas en réponse directe à une demande du client, ou une action entreprise par un client peut indirectement impliquer un second client. Des exemples de telles actions incluent la suppression à expiration, le renouvellement automatique à expiration, et la coordination de transfert ; d'autres types d'informations de service PEUVENT être définis au titre de la politique du serveur. Les messages de service DEVRAIENT être créés pour des clients passifs affectés par une action sur un objet. Les messages de service PEUVENT aussi être créés pour des clients actifs qui demandent une action sur un objet, bien que de tels messages NE DOIVENT PAS remplacer la réponse normale de protocole à la demande. Par exemple, des actions <transfer> DEVRAIENT être rapportées au client qui a l'autorité pour approuver ou rejeter une demande de transfert. D'autres méthodes de notification d'action serveur-client, comme un rapport hors ligne, sont aussi possibles et sortent du domaine d'application de la présente spécification.

Les files d'attente de messages peuvent consommer les ressources du serveur si les clients ne récupèrent et n'acquittent pas les messages de façon régulière. Les serveurs PEUVENT mettre en œuvre d'autres mécanismes pour sortir de file d'attente et livrer les messages si les besoins de la maintenance de file d'attente excèdent les limites de consommation des ressources du serveur. Les opérateurs de serveur DEVRAIENT considérer les facteurs de sensibilité au temps et de gestion de ressource lors du choix d'une méthode de livraison des informations de service parce que certains types de message peuvent être raisonnablement livrés en utilisant des méthodes hors protocole qui requièrent moins de ressources du serveur.

Certaines des informations retournées en réponse à une commande <poll> peuvent être spécifiques de l'objet, de sorte que certains éléments fils de la réponse <poll> PEUVENT être spécifiés en utilisant le cadre d'extension EPP. La commande <poll> DOIT être représentée comme un élément vide sans élément fils. Un attribut "op" d'une valeur de "req" est EXIGÉ pour restituer le premier message de la file d'attente de messages du serveur. Un attribut "op" (d'une valeur de "ack") et un attribut "msgID" (dont la valeur correspond à la valeur de l'attribut "id" copié de l'élément <msg> dans le message dont on accuse

réception) sont EXIGÉS pour accuser réception d'un message.

Exemple de commande <poll> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <poll op="req"/>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Le code de résultat retourné note qu'un message a été sorti de la file d'attente et retourné en réponse à une commande <poll>.

Exemple de réponse <poll> avec des informations spécifiques de l'objet :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Commande achevée avec succès ; accusé de réception de sortie de file d'attente</msg>
S:    </result>
S:    <msgQ count="5" id="12345">
S:      <qDate>2000-06-08T22:00:00.0Z</qDate>
S:      <msg>Transfert demandé.</msg>
S:    </msgQ>
S:    <resData>
S:      <obj:trnData
S:       xmlns:obj="urn:ietf:params:xml:ns:obj-1.0">
S:        <obj:name>example.com</obj:name>
S:        <obj:trStatus>pending</obj:trStatus>
S:        <obj:reID>ClientX</obj:reID>
S:        <obj:reDate>2000-06-08T22:00:00.0Z</obj:reDate>
S:        <obj:acID>ClientY</obj:acID>
S:        <obj:acDate>2000-06-13T22:00:00.0Z</obj:acDate>
S:        <obj:exDate>2002-09-08T22:00:00.0Z</obj:exDate>
S:      </obj:trnData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Un client DOIT accuser réception de chaque réponse pour sortir le message de file d'attente et rendre les messages suivants disponibles pour restitution.

Exemple de commande d'accusé de réception <poll> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <poll op="ack" msgID="12345"/>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

Une réponse d'accusé de réception <poll> note l'identifiant du message qui a été acquitté et le nombre de messages restants dans la file.

Exemple de réponse d'accusé de réception <poll> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <msgQ count="4" id="12345"/>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Les messages de service peuvent aussi être retournés sans information d'objet.

Exemple de réponse <poll> avec un contenu de message mixte et sans information spécifique de l'objet :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1301">
S:      <msg>Commande achevée avec succès ; accusé de réception de sortie de file d'attente</msg>
S:    </result>
S:    <msgQ count="4" id="12346">
S:      <qDate>2000-06-08T22:10:00.0Z</qDate>
S:      <msg lang="fr">Solde du compte faible.
S:        <limit>100</limit><bal>5</bal>
S:      </msg>
S:    </msgQ>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

Le code de résultat et le message retournés sont utilisés pour noter une file d'attente vide de message au serveur.

Exemple de réponse <poll> pour noter une file d'attente vide de message :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1300">
S:      <msg>Commande achevée avec succès ; pas de message</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <poll> est utilisée pour découvrir et restituer au client les messages de service provenant d'un serveur. Cette action DEVRAIT être limitée aux clients autorisés ; mettre en file d'attente les messages de service et limiter l'accès à la file d'attente client par client est RECOMMANDÉ.

2.9.2.4 Commande d'interrogation EPP <transfer>

La commande EPP <transfer> fournit une opération d'interrogation qui permet à un client de déterminer en temps réel l'état en

cours et achevé des demandes de transferts. Les éléments nécessaires pour identifier un objet qui est le sujet d'une demande de transfert sont spécifiques de l'objet, de sorte que les éléments fils de la commande d'interrogation <transfer> sont spécifiés en utilisant le cadre d'extension EPP. En plus des éléments standard de commande EPP, la commande <transfer> contient un attribut "op" d'une valeur de "query" et les éléments fils suivants :

- Un élément <obj:transfer> spécifique de l'objet qui identifie l'objet dont l'état de transfert est demandé.

L'état de transfert est normalement considéré comme une information sensible par les clients impliqués dans l'opération. Les transpositions d'objet DOIVENT fournir des dispositifs pour restreindre les interrogations de transfert aux clients autorisés, comme d'exiger des informations d'autorisation au titre de la demande .

Exemple de commande d'interrogation <transfer> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="query">
C:      <obj:transfer xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <!-- Éléments spécifiques de l'objet. -->
C:      </obj:transfer>
C:    </transfer>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande d'interrogation <transfer> a été traitée avec succès, un serveur DOIT répondre avec un élément EPP <resData> qui DOIT contenir un élément fils qui identifie l'espace de nom de l'objet. Les éléments fils de l'élément <resData> sont spécifiques de l'objet, mais ils DOIVENT inclure des éléments qui identifient l'objet, l'état du transfert, l'identifiant du client qui a demandé le transfert, la date et l'heure à laquelle la demande a été faite, l'identifiant du client qui est autorisé à agir sur la demande, la date et l'heure à laquelle une action est attendue, et une date et heure FACULTATIVES qui notent les changements de la période de validité de l'objet (si applicable) qui résultent du transfert.

Exemple de réponse d'interrogation <transfer> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <resData>
S:      <obj:trnData xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <obj:name>example</obj:name>
S:        <obj:trStatus>pending</obj:trStatus>
S:        <obj:reID>ClientX</obj:reID>
S:        <obj:reDate>2000-06-08T22:00:00.0Z</obj:reDate>
S:        <obj:acID>ClientY</obj:acID>
S:        <obj:acDate>2000-06-13T22:00:00.0Z</obj:acDate>
S:        <obj:exDate>2002-09-08T22:00:00.0Z</obj:exDate>
S:      </obj:trnData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <transfer> fournit une opération d'interrogation qui permet à un client de déterminer en temps réel l'état en cours et achevé des demandes de transfert. Cette action DEVRAIT être limitée aux clients autorisés ; restreindre les interrogations aux clients demandeur et répondant est RECOMMANDÉ. Le transfert d'objet PEUT être indisponible ou limité par des politiques spécifiques de l'objet.

2.9.3 Commandes de transformation d'objet

EPP fournit cinq commandes pour transformer les objets : <create> pour créer une instance d'un objet avec un serveur, <delete> pour supprimer une instance d'objet d'un serveur, <renew> pour étendre la période de validité d'un objet, <transfer> pour gérer des changements du parrainage d'un objet par un client, et <update> pour changer les informations associées à un objet.

2.9.3.1 Commande EPP <create>

La commande EPP <create> est utilisée pour créer une instance d'un objet. Un objet peut être créé pour une période indéfinie, ou pour une période de validité spécifique. La transposition EPP pour un objet DOIT décrire l'état d'un objet par rapport au temps afin d'inclure le comportement attendu du client et du serveur si une période de validité est utilisée.

Les éléments nécessaires pour identifier un objet et les attributs qui lui sont associés sont spécifiques de l'objet, de sorte que les éléments fils de la commande <create> sont spécifiés en utilisant le cadre d'extension EPP. En plus des éléments standard de commande EPP, la commande <create> contient les éléments fils suivants :

- Un élément <obj:create> spécifique de l'objet qui identifie l'objet à créer et les éléments qui sont nécessaires pour créer l'objet.

Exemple de commande <create> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <create>
C:      <obj:create xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <!-- Éléments spécifiques de l'objet. -->
C:      </obj:create>
C:    </create>
C:    <clTRID>ABC-12345</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <create> a été traitée avec succès, un serveur PEUT répondre par un élément EPP <resData> qui DOIT contenir un élément fils qui identifie l'espace de noms de l'objet. Les éléments fils de l'élément <resData> sont spécifiques de l'objet.

Exemple de réponse <create> avec <resData> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <resData>
S:      <obj:creData xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <!-- Éléments spécifiques de l'objet. -->
S:      </obj:creData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12345</clTRID>
S:      <svTRID>54321-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <create> est utilisée pour créer une instance d'un objet. Cette action DEVRAIT être limitée aux clients autorisés et PEUT être restreinte selon le client.

2.9.3.2 Commande EPP <delete>

La commande EPP <delete> est utilisée pour supprimer une instance d'un objet existant. Les éléments nécessaires pour

identifier un objet sont spécifiques de l'objet, de sorte que les éléments fils de la commande <delete> sont spécifiés en utilisant le cadre d'extension EPP. En plus des éléments standard de commande EPP, la commande <delete> contient les éléments fils suivants :

- Un élément <obj:delete> spécifique de l'objet qui identifie l'objet à supprimer.

Exemple de commande <delete> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <delete>
C:      <obj:delete xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <!-- Éléments spécifiques de l'objet. -->
C:      </obj:delete>
C:    </delete>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <delete> a été traitée avec succès, un serveur PEUT répondre par un élément EPP <resData> qui DOIT contenir un élément fils qui identifie l'espace de noms de l'objet. Les éléments fils de l'élément <resData> sont spécifiques de l'objet.

Exemple de réponse <delete> sans <resData> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <delete> est utilisée pour supprimer une instance d'un objet existant. Cette action DEVRAIT être limitée aux clients autorisés ; restreindre cette action au client parrain est RECOMMANDÉ.

2.9.3.3 Commande EPP <renew>

La commande EPP <renew> est utilisée pour étendre la période de validité d'un objet existant. Les éléments nécessaires pour identifier et étendre la période de validité d'un objet sont spécifiques de l'objet, de sorte que les éléments fils de la commande <renew> sont spécifiés en utilisant le cadre d'extension EPP. En plus des éléments standard de commande EPP, la commande <renew> contient les éléments fils suivants :

- Un élément <obj:renew> spécifique de l'objet qui identifie l'objet à renouveler et les éléments requis pour étendre la période de validité de l'objet.

Exemple de commande <renew> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <renew>
C:      <obj:renew xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <!-- Éléments spécifiques de l'objet. -->
C:      </obj:renew>
C:    </renew>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <renew> a été traitée avec succès, un serveur PEUT répondre avec un élément EPP <resData> qui DOIT contenir un élément fils qui identifie l'espace de noms de l'objet. Les éléments fils de l'élément <resData> sont spécifiques de l'objet.

Exemple de réponse <renew> avec <resData> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <resData>
S:      <obj:renData xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <!-- Éléments spécifiques de l'objet. -->
S:      </obj:renData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <renew> est utilisée pour étendre la période de validité d'un objet existant. Cette action DEVRAIT être limitée aux clients autorisés ; restreindre cette action au client parrain est RECOMMANDÉ. Le renouvellement d'objet PEUT être indisponible ou limité par des politiques spécifiques de l'objet.

2.9.3.4 Commande EPP <transfer>

La commande EPP <transfer> est utilisée pour gérer les changements de parrainage de client d'un objet existant. Les clients peuvent initier une demande de transfert, annuler une demande de transfert, approuver une demande de transfert, et rejeter une demande de transfert en utilisant l'attribut de commande "op".

Un client qui souhaite assumer le parrainage d'un objet connu d'un autre client utilise la commande <transfer> avec la valeur de l'attribut "op" réglée à "request". Une fois que le transfert a été demandé, le même client peut annuler la demande en utilisant une commande <transfer> avec la valeur de l'attribut "op" réglée à "cancel". Une demande d'annulation de transfert DOIT être envoyée au serveur avant que le client parrain actuel ait approuvé ou rejeté la demande de transfert et avant que le serveur traite automatiquement la demande à cause de l'inactivité du client répondant.

Une fois qu'une demande de transfert a été reçue par le serveur, celui-ci DOIT notifier au client parrain actuel le transfert demandé soit en mettant en file d'attente un message de service à restituer via la commande <poll>, soit en utilisant un mécanisme hors bande pour informer le client de la demande. L'état actuel d'une commande <transfer> en cours pour tout objet peut être trouvé en utilisant la commande d'interrogation <transfer>. Les messages de service de transfert DOIVENT inclure les éléments spécifiques de l'objet spécifiés pour les réponses de commande <transfer>.

Le client parrain actuel PEUT explicitement approuver ou rejeter la demande de transfert. Le client peut approuver la demande en utilisant la commande <transfer> avec la valeur de l'attribut "op" réglée à "approve". Le client peut rejeter la demande en utilisant une commande <transfer> avec la valeur de l'attribut "op" réglée à "reject".

Un serveur PEUT automatiquement approuver ou rejeter toutes les demandes de transfert qui ne sont pas explicitement approuvées ou rejetées par le client parrain actuel dans un délai fixé. Le délai d'attente d'une action explicite et le comportement par défaut du serveur sont des affaires locales non spécifiées par EPP, mais ils DEVRAIENT être documentés dans un document de profil spécifique du serveur qui décrit le comportement par défaut du serveur pour les informations client.

Les objets éligibles au transfert DOIVENT avoir les informations d'autorisation associées qui DOIVENT être fournies pour achever une commande <transfer>. Le type d'informations d'autorisation requises est spécifique de l'objet ; des mécanismes de mot de passe ou plus complexes fondés sur le chiffrement à clé publique sont courants.

Les éléments nécessaires pour identifier et achever le transfert d'un objet sont spécifiques de l'objet, de sorte que les éléments

fils de la commande <transfer> sont spécifiés en utilisant le cadre d'extension EPP. En plus des éléments standard de commande EPP, la commande <transfer> contient les éléments fils suivants :

- Un élément <obj:transfer> spécifique de l'objet qui identifie l'objet à transférer et les éléments qui sont requis pour traiter la commande de transfert.

Exemple de commande <transfer> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <transfer op="request">
C:      <obj:transfer xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <!-- Éléments spécifiques de l'objet. -->
C:      </obj:transfer>
C:    </transfer>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <transfer> a été traitée avec succès, un serveur DOIT répondre par un élément EPP <resData> qui DOIT contenir un élément fils qui identifie l'espace de noms de l'objet. Les éléments fils de l'élément <resData> sont spécifiques de l'objet, mais ils DOIVENT inclure les éléments qui identifient l'objet, l'état du transfert, l'identifiant du client qui a demandé le transfert, la date et l'heure de la demande, l'identifiant du client qui est autorisé à agir sur la demande, la date et l'heure à laquelle une action est attendue, et une date et heure FACULTATIVE qui note les changements de la période de validité de l'objet (si applicable) qui surviennent par suite du transfert.

Exemple de réponse <transfer> avec <resData> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1001">
S:      <msg>Commande achevée avec succès; action en cours</msg>
S:    </result>
S:    <resData>
S:      <obj:trnData xmlns:obj="urn:ietf:params:xml:ns:obj">
S:        <obj:name>example</obj:name>
S:        <obj:trStatus>pending</obj:trStatus>
S:        <obj:reID>ClientX</obj:reID>
S:        <obj:reDate>2000-06-08T22:00:00.0Z</obj:reDate>
S:        <obj:acID>ClientY</obj:acID>
S:        <obj:acDate>2000-06-13T22:00:00.0Z</obj:acDate>
S:        <obj:exDate>2002-09-08T22:00:00.0Z</obj:exDate>
S:      </obj:trnData>
S:    </resData>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <transfer> est utilisée pour gérer les changements de client parrain d'un objet existant. Cette action DEVRAIT être limitée aux clients autorisés ; restreindre les demandes <transfer> à un client autre que le client parrain actuel, les demandes d'approbation de <transfer> au client parrain actuel, et les demandes d'annulation de <transfer> au client demandeur d'origine est RECOMMANDÉ. Le transfert d'objet PEUT être indisponible ou limité par des politiques spécifiques de l'objet.

2.9.3.5 Commande EPP <update>

La commande EPP <update> est utilisée pour changer les informations associées à un objet existant. Les éléments nécessaires pour identifier et modifier un objet sont spécifiques de l'objet, de sorte que les éléments fils de la commande <update> sont

spécifiés en utilisant le cadre d'extension EPP. En plus des éléments standard de commande EPP, la commande <update> contient les éléments fils suivants :

- Élément spécifique de l'objet <obj:update> qui identifie l'objet à mettre à jour et les éléments qui sont requis pour modifier l'objet. Les éléments spécifiques de l'objet DOIVENT identifier les valeurs à ajouter, les valeurs à retirer, ou les valeurs à changer.

Exemple de commande <update> :

```
C:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
C:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
C:  <command>
C:    <update>
C:      <obj:update xmlns:obj="urn:ietf:params:xml:ns:obj">
C:        <!-- Éléments spécifiques de l'objet. -->
C:      </obj:update>
C:    </update>
C:    <clTRID>ABC-12346</clTRID>
C:  </command>
C:</epp>
```

Lorsque une commande <update> a été traitée avec succès, un serveur PEUT répondre avec un élément EPP <resData> qui DOIT contenir un élément fils qui identifie l'espace de noms de l'objet. Les éléments fils de l'élément <resData> sont spécifiques de l'objet.

Exemple de réponse <update> sans <resData> :

```
S:<?xml version="1.0" encoding="UTF-8" standalone="no"?>
S:<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
S:  <response>
S:    <result code="1000">
S:      <msg>Commande achevée avec succès</msg>
S:    </result>
S:    <trID>
S:      <clTRID>ABC-12346</clTRID>
S:      <svTRID>54322-XYZ</svTRID>
S:    </trID>
S:  </response>
S:</epp>
```

La commande EPP <update> est utilisée pour changer les informations associées à un objet existant. Cette action DEVRAIT être limitée aux clients autorisés ; restreindre cette action au client parrain est RECOMMANDÉ.

3. Codes de résultat

Les codes de résultat EPP se fondent sur la théorie des codes de réponse décrite au paragraphe 4.2.1 de la [RFC5321]. EPP utilise quatre chiffres décimaux pour décrire le succès ou l'échec de chaque commande EPP. Chacun des chiffres de la réponse a une signification particulière.

Le premier chiffre note le succès ou l'échec de la commande. Le second chiffre note la catégorie de réponse, comme la syntaxe ou la sécurité de la commande. Les troisièmes et quatrièmes chiffres donnent le détail explicite de la réponse au sein de chaque catégorie de réponse.

Il y a deux valeurs pour le premier chiffre du code de réponse :
1yzz : Réponse d'achèvement positif. La commande a été acceptée et traitée sans erreur par le système.
2yzz : Réponse d'achèvement négatif. La commande n'a pas été acceptée, et l'action demandée ne s'est pas produite.

Le second chiffre groupe les réponses en six catégories spécifiques :
x0zz : Syntaxe du protocole
x1zz : Règles spécifiques de la mise en œuvre
x2zz : Sécurité
x3zz : Gestion des données

x4zz : Système serveur
x5zz : Gestion de la connexion

Le troisième et le quatrième chiffre fournissent le détail de la réponse au sein de la catégorie définie par le premier et le second chiffre. La liste complète des codes de résultat valides figure ci-dessous et dans le schéma normatif.

Chaque réponse EPP DOIT inclure un code de résultat et une description lisible par l'homme du code de résultat. Le langage utilisé pour représenter la description PEUT être identifié en utilisant une instance de l'attribut "lang" au sein de l'élément <msg>. Si il n'est pas spécifié, le langage par défaut est l'anglais, identifié par "en". Une description de la structure des valeurs valides de l'attribut "lang" figure dans la [RFC4646].

Le texte de réponse PEUT être traduit dans d'autres langues, mais la traduction DOIT préserver la signification du code décrite ici. Les valeurs de code de réponse NE DOIVENT PAS être changées lors de la traduction du texte.

Le texte des réponses dans le tableau ci-dessous est mis entre guillemets pour marquer clairement le début et la fin de chaque chaîne de réponse. Les guillemets NE DOIVENT PAS être utilisés pour délimiter ces chaînes lors du retour du texte de réponse via le protocole.

Réponses de succès d'achèvement de commande :

Code	Texte de réponse en français	Commentaire
1000	"Commande achevée avec succès"	Code de réponse normal pour l'achèvement réussi d'une commande qui n'est pas visé par un autre code de réponse de la série 1xxx.
1001	"Commande achevée avec succès ; action en cours"	Ce code de réponse DOIT être retourné lorsqu'une réponse à une commande exige une activité hors ligne avant que l'action demandée puisse se faire. Voir à la Section 2 la description des autres exigences de traitement.
1300	"Commande achevée avec succès ; pas de message"	Ce code de réponse DOIT être retourné pour répondre à une commande de demande <poll> et que la file d'attente de message du serveur est vide.
1301	"Commande achevée avec succès ; accuser réception pour sortie"	Ce code de réponse DOIT être retourné pour répondre à une commande de demande <poll> et qu'un message a été restitué de la file d'attente de messages du serveur.
1500	"Commande achevée avec succès ; fin de session"	Ce code de réponse DOIT être retourné pour répondre à une commande <logout> réussie.

Réponse d'erreur de commande :

Code	Texte de réponse en français	Commentaire
2000	"Commande inconnue"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit un élément de commande non défini dans EPP.
2001	"Erreur de syntaxe de commande"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit un élément de commande mal formé.
2002	"Erreur d'utilisation de commande"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit un élément de commande bien formé mais que la commande ne peut s'exécuter à cause d'une erreur de séquençage ou de contexte. Par exemple, une commande <logout> ne peut être exécutée sans avoir d'abord achevé une commande <login>.
2003	"Paramètre exigé manquant"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande pour laquelle une valeur de paramètre exigé n'a pas été fournie.
2004	"Erreur de gamme de valeur de paramètre"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit un paramètre de commande dont la valeur est hors de la gamme des valeurs spécifiée par le protocole. La valeur d'erreur DEVRAIT être retournée via un élément <value> dans la réponse EPP.
2005	"Erreur de syntaxe de valeur de paramètre"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande contenant un paramètre dont la valeur est mal formée. La valeur d'erreur DEVRAIT être retournée via un élément <value> dans la réponse EPP.
2100	"Version de protocole non mise en œuvre"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit un élément de commande spécifiant une version de protocole non mise en œuvre par le serveur.
2101	"Commande non mise en œuvre"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit un élément de commande EPP valide qui n'est pas mis en œuvre par le serveur. Par exemple, une commande <transfer> peut n'être pas mise en œuvre pour certains types d'objet.
2102	"Option non mise en œuvre"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit un élément de commande EPP valide qui contient une option du protocole qui n'est pas mise en

		œuvre par le serveur.
2103	"Extension non mise en œuvre"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit un élément de commande EPP valide qui contient une extension de commande du protocole qui n'est pas mise en œuvre par le serveur.
2104	"Échec de facturation"	Ce code de réponse DOIT être retourné lorsque un serveur tente d'exécuter une opération facturable et que la commande ne peut pas être achevée à cause d'un échec de la facturation du client.
2105	"Objet non renouvelable"	Ce code de réponse DOIT être retourné lorsque un client tente de renouveler un objet qui n'est pas éligible au renouvellement selon la politique du serveur.
2106	"Objet non transférable"	Ce code de réponse DOIT être retourné lorsque un client tente de transférer un objet qui n'est pas éligible au transfert selon la politique du serveur.
2200	"Erreur d'authentification"	Ce code de réponse DOIT être retourné lorsque un serveur note une erreur en validant les accréditifs du client.
2201	"Erreur d'autorisation"	Ce code de réponse DOIT être retourné lorsque un serveur note une erreur d'autorisation de client en exécutant une commande. Ce code est utilisé pour noter qu'un client manque des privilèges pour exécuter la commande demandée.
2202	"Information d'autorisation invalides"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit des informations d'autorisation de commande invalides en tentant de confirmer l'autorisation d'exécuter une commande. Ce code est utilisé pour noter qu'un client a les privilèges requis pour exécuter la commande demandée, mais que les informations d'autorisation fournies par le client ne correspondent pas aux informations d'autorisation archivées par le serveur.
2300	"Transfert d'objet en cours"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande de transfert d'un objet qui est en cours de transfert suite à une demande de transfert antérieure.
2301	"L'objet n'est pas en cours de transfert"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande de confirmation, rejet, ou annulation du transfert d'un objet lorsque aucune commande n'a été faite de transférer l'objet.
2302	"Objet existant"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande de créer un objet qui existe déjà dans le répertoire.
2303	"Objet non existant"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande d'interroger ou de transformer un objet qui n'existe pas dans le répertoire.
2304	"Le statut de l'objet interdit l'opération"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande de transformer un objet qui ne peut être achevée à cause de la politique du serveur ou de pratiques commerciales. Par exemple, un serveur peut interdire des commandes <transfer> selon des termes et conditions qui relèvent de la politique locale, ou le serveur pourrait avoir reçu une commande <delete> pour un objet dont l'état interdit la suppression.
2305	"L'association d'objets interdit l'opération"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande de transformer un objet qui ne peut être achevée à cause d'une dépendance à d'autres objets qui sont associés à l'objet cible. Par exemple, un serveur peut interdire des commandes <delete> alors qu'un objet a des associations actives avec d'autres objets.
2306	"Erreur de politique de valeur de paramètre"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande contenant une valeur de paramètre qui est syntaxiquement valide mais sémantiquement invalide à cause de la politique locale. Par exemple, le serveur peut prendre en charge un sous ensemble d'une gamme de valeurs de protocole valides. La valeur d'erreur DEVRAIT être retournée via un élément <value> dans la réponse EPP.
2307	"Service d'objet non mis en œuvre"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande d'opérer sur un service d'objet qui n'est pas pris en charge par le serveur.
2308	"Violation de la politique de gestion des données"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande dont l'exécution résulte en une violation de la politique de gestion des données du serveur. Par exemple, supprimer toutes les valeurs d'attribut ou les associations d'objet à partir d'un objet pourrait être une violation de la politique de gestion des données d'un serveur.
2400	"Échec de commande"	Ce code de réponse DOIT être retourné lorsque un serveur ne peut exécuter une commande du fait d'une erreur interne du serveur sans relation avec le protocole. L'échec peut être temporaire. Le serveur DOIT garder actives toutes les sessions en cours.
2500	"Échec de commande ; le serveur ferme la connexion"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande qui ne peut être achevée à cause d'une erreur interne du serveur sans relation

		avec le protocole. L'échec n'est pas temporaire et va causer aussi l'échec des autres commandes. Le serveur DOIT clore les sessions actives et clore la connexion existante.
2501	"Erreur d'authentification ; le serveur ferme la connexion"	Ce code de réponse DOIT être retourné lorsque un serveur note une erreur en validant les accréditifs du client et qu'une limite définie par le serveur du nombre d'échecs admissibles a été dépassée. Le serveur DOIT clore la connexion existante.
2502	"Limite de session dépassée ; le serveur ferme la connexion"	Ce code de réponse DOIT être retourné lorsque un serveur reçoit une commande <login> et que la commande ne peut être achevée parce que le client a excédé une limite définie par le système sur le nombre de sessions que le client peut établir. Il est possible d'établir une session en terminant des sessions existantes non utilisées et en fermant les connexions inactives.

4. Syntaxe formelle

EPP est spécifié en notation de schéma XML. La syntaxe formelle présentée ici est une représentation de schéma complète de EPP qui convient pour les instances de validation automatique de XML EPP.

Deux schémas sont présentés ici. Le premier schéma est le schéma EPP de base. Le second schéma définit des éléments et des structures qui peuvent être utilisés par les deux schémas EPP de base et de transposition d'objet. Les étiquettes DÉBUT et FIN ne font pas partie du schéma ; elles sont utilisées pour noter le début et la fin du schéma pour les besoins d'enregistrement d'URI.

4.1 Schéma de base

DÉBUT

```
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="urn:ietf:params:xml:ns:epp-1.0"
      xmlns:epp="urn:ietf:params:xml:ns:epp-1.0"
      xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
      xmlns="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified">

<!--  Importation des types d'élément communs.   -->
 <import namespace="urn:ietf:params:xml:ns:eppcom-1.0"/>

  <annotation>
```

```
    <documentation>
      Extensible Provisioning Protocol v1.0 schema.
    </documentation>
  </annotation>

<!--  Toute instance de XML EPP doit commencer par cet élément.  -->

  <element name="epp" type="epp:eppType"/>

<!--  Une instance XML EPP doit contenir un salut, un hello, une commande, une réponse, ou une extension.  -->

  <complexType name="eppType">
    <choice>
      <element name="greeting" type="epp:greetingType"/>
      <element name="hello"/>
      <element name="command" type="epp:commandType"/>
      <element name="response" type="epp:responseType"/>
      <element name="extension" type="epp:extAnyType"/>
    </choice>
  </complexType>

<!--  Un salut est envoyé par un serveur en réponse à une connexion ou <hello> d'un client.  -->
  <complexType name="greetingType">
    <sequence>
      <element name="svID" type="epp:sIDType"/>
      <element name="svDate" type="dateTime"/>
      <element name="svcMenu" type="epp:svcMenuType"/>
      <element name="dcp" type="epp:dcpType"/>
    </sequence>
  </complexType>

<!--  Les identifiants de serveur sont des chaînes avec des restrictions de longueur minimum et maximum.  -->
  <simpleType name="sIDType">
    <restriction base="normalizedString">
      <minLength value="3"/>
      <maxLength value="64"/>
    </restriction>
  </simpleType>

<!--  Un salut de serveur identifie les services d'objet disponibles.  -->
  <complexType name="svcMenuType">
    <sequence>
      <element name="version" type="epp:versionType"
       maxOccurs="unbounded"/>
      <element name="lang" type="language"
       maxOccurs="unbounded"/>
      <element name="objURI" type="anyURI"
       maxOccurs="unbounded"/>
      <element name="svcExtension" type="epp:extURIType"
       minOccurs="0"/>
    </sequence>
  </complexType>

<!--  Types de politique de collecte des données.  -->
  <complexType name="dcpType">
    <sequence>
      <element name="access" type="epp:dcpAccessType"/>
      <element name="statement" type="epp:dcpStatementType"
       maxOccurs="unbounded"/>
      <element name="expiry" type="epp:dcpExpiryType"
       minOccurs="0"/>
    </sequence>
```

```
</complexType>

<complexType name="dcpAccessType">
 <choice>
  <element name="all"/>
  <element name="none"/>
  <element name="null"/>
  <element name="other"/>
  <element name="personal"/>
  <element name="personalAndOther"/>
 </choice>
</complexType>

<complexType name="dcpStatementType">
 <sequence>
  <element name="purpose" type="epp:dcpPurposeType"/>
  <element name="recipient" type="epp:dcpRecipientType"/>
  <element name="retention" type="epp:dcpRetentionType"/>
 </sequence>
</complexType>

<complexType name="dcpPurposeType">
 <sequence>
  <element name="admin"
   minOccurs="0"/>
  <element name="contact"
   minOccurs="0"/>
  <element name="other"
   minOccurs="0"/>
  <element name="prov"
   minOccurs="0"/>
 </sequence>
</complexType>

<complexType name="dcpRecipientType">
 <sequence>
  <element name="other"
   minOccurs="0"/>
  <element name="ours" type="epp:dcpOursType"
   minOccurs="0" maxOccurs="unbounded"/>
  <element name="public"
   minOccurs="0"/>
  <element name="same"
   minOccurs="0"/>
  <element name="unrelated"
   minOccurs="0"/>
 </sequence>
</complexType>

<complexType name="dcpOursType">
 <sequence>
  <element name="recDesc" type="epp:dcpRecDescType"
   minOccurs="0"/>
 </sequence>
</complexType>

<simpleType name="dcpRecDescType">
 <restriction base="token">
  <minLength value="1"/>
  <maxLength value="255"/>
 </restriction>
</simpleType>
```

```
  <complexType name="dcpRetentionType">
   <choice>
    <element name="business"/>
    <element name="indefinite"/>
    <element name="legal"/>
    <element name="none"/>
    <element name="stated"/>
   </choice>
  </complexType>

  <complexType name="dcpExpiryType">
   <choice>
    <element name="absolute" type="dateTime"/>
    <element name="relative" type="duration"/>
   </choice>
  </complexType>

 <!--  Extension framework types.  -->
  <complexType name="extAnyType">
   <sequence>
    <any namespace="##other"
     maxOccurs="unbounded"/>
   </sequence>
  </complexType>

  <complexType name="extURIType">
   <sequence>
    <element name="extURI" type="anyURI"
     maxOccurs="unbounded"/>
   </sequence>
  </complexType>

 <!--  Un numéro de version EPP est une paire de nombres décimaux séparés par des points.  -->
  <simpleType name="versionType">
   <restriction base="token">
    <pattern value="[1-9]+\.[0-9]+"/>
    <enumeration value="1.0"/>
   </restriction>
  </simpleType>

 <!--  Types de commandes.  -->
  <complexType name="commandType">
   <sequence>
    <choice>
     <element name="check" type="epp:readWriteType"/>
     <element name="create" type="epp:readWriteType"/>
     <element name="delete" type="epp:readWriteType"/>
     <element name="info" type="epp:readWriteType"/>
     <element name="login" type="epp:loginType"/>
     <element name="logout"/>
     <element name="poll" type="epp:pollType"/>
     <element name="renew" type="epp:readWriteType"/>
     <element name="transfer" type="epp:transferType"/>
     <element name="update" type="epp:readWriteType"/>
    </choice>
    <element name="extension" type="epp:extAnyType"
     minOccurs="0"/>
    <element name="clTRID" type="epp:trIDStringType"
     minOccurs="0"/>
   </sequence>
  </complexType>
```

```
<!--   Commande <login>.   -->
 <complexType name="loginType">
  <sequence>
   <element name="clID" type="eppcom:clIDType"/>
   <element name="pw" type="epp:pwType"/>
   <element name="newPW" type="epp:pwType"
    minOccurs="0"/>
   <element name="options" type="epp:credsOptionsType"/>
   <element name="svcs" type="epp:loginSvcType"/>
  </sequence>
 </complexType>

 <complexType name="credsOptionsType">
  <sequence>
   <element name="version" type="epp:versionType"/>
   <element name="lang" type="language"/>
  </sequence>
 </complexType>

 <simpleType name="pwType">
  <restriction base="token">
   <minLength value="6"/>
   <maxLength value="16"/>
  </restriction>
 </simpleType>

 <complexType name="loginSvcType">
  <sequence>
   <element name="objURI" type="anyURI"
    maxOccurs="unbounded"/>
   <element name="svcExtension" type="epp:extURIType"
    minOccurs="0"/>
  </sequence>
 </complexType>

<!--   Commande <poll>.   -->
 <complexType name="pollType">
  <attribute name="op" type="epp:pollOpType"
   use="required"/>

  <attribute name="msgID" type="token"/>
 </complexType>

 <simpleType name="pollOpType">
  <restriction base="token">
   <enumeration value="ack"/>
   <enumeration value="req"/>
  </restriction>
 </simpleType>

<!--   Commande <transfer>. Elle est spécifique de l'objet, et utilise des attributs pour identifier l'opération demandée.   -->
 <complexType name="transferType">
  <sequence>
   <any namespace="##other"/>
  </sequence>
  <attribute name="op" type="epp:transferOpType"
   use="required"/>
 </complexType>

 <simpleType name="transferOpType">
  <restriction base="token">
```

```
        <enumeration value="approve"/>
        <enumeration value="cancel"/>
        <enumeration value="query"/>
        <enumeration value="reject"/>
        <enumeration value="request"/>
      </restriction>
    </simpleType>

<!-- Toutes les autres commandes centrées sur l'objet. EPP ne spécifie pas la syntaxe et la sémantique des éléments de
commande centrés sur l'objet. Les éléments DOIVENT être décrits en détail dans un autre schéma spécifique de l'objet. -->
    <complexType name="readWriteType">
      <sequence>
        <any namespace="##other"/>
      </sequence>
    </complexType>

    <complexType name="trIDType">
      <sequence>
        <element name="clTRID" type="epp:trIDStringType"
         minOccurs="0"/>
        <element name="svTRID" type="epp:trIDStringType"/>
      </sequence>
    </complexType>

    <simpleType name="trIDStringType">
      <restriction base="token">
        <minLength value="3"/>
        <maxLength value="64"/>
      </restriction>
    </simpleType>

<!-- Types de réponse. -->
    <complexType name="responseType">
      <sequence>
        <élément name="result" type="epp:resultType"
         maxOccurs="unbounded"/>
        <element name="msgQ" type="epp:msgQType"
         minOccurs="0"/>
        <element name="resData" type="epp:extAnyType"
         minOccurs="0"/>
        <element name="extension" type="epp:extAnyType"
         minOccurs="0"/>
        <element name="trID" type="epp:trIDType"/>
      </sequence>
    </complexType>

    <complexType name="resultType">
      <sequence>
        <element name="msg" type="epp:msgType"/>
        <choice minOccurs="0" maxOccurs="unbounded">
          <element name="value" type="epp:errValueType"/>
          <element name="extValue" type="epp:extErrValueType"/>
        </choice>
      </sequence>
      <attribute name="code" type="epp:resultCodeType"
       use="required"/>
    </complexType>

    <complexType name="errValueType" mixed="true">
      <sequence>
        <any namespace="##any" processContents="skip"/>
      </sequence>
```

```
      <anyAttribute namespace="##any" processContents="skip"/>
    </complexType>

    <complexType name="extErrValueType">
      <sequence>
        <element name="value" type="epp:errValueType"/>
        <element name="reason" type="epp:msgType"/>
      </sequence>
    </complexType>

    <complexType name="msgQType">
      <sequence>
        <element name="qDate" type="dateTime"
         minOccurs="0"/>
        <element name="msg" type="epp:mixedMsgType"
         minOccurs="0"/>
      </sequence>
      <attribute name="count" type="unsignedLong"
       use="required"/>
      <attribute name="id" type="eppcom:minTokenType"
       use="required"/>
    </complexType>

    <complexType name="mixedMsgType" mixed="true">
      <sequence>
        <any processContents="skip"
         minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="lang" type="language"
       default="en"/>
    </complexType>

  <!-- Texte lisible par l'homme, peut être exprimé dans des langages autres que l'anglais. -->
    <complexType name="msgType">
      <simpleContent>
        <extension base="normalizedString">
          <attribute name="lang" type="language"
           default="en"/>
        </extension>
      </simpleContent>
    </complexType>

  <!-- Codes de résultat EPP. -->
    <simpleType name="resultCodeType">
      <restriction base="unsignedShort">
        <enumeration value="1000"/>
        <enumeration value="1001"/>
        <enumeration value="1300"/>
        <enumeration value="1301"/>
        <enumeration value="1500"/>
        <enumeration value="2000"/>
        <enumeration value="2001"/>
        <enumeration value="2002"/>
        <enumeration value="2003"/>
        <enumeration value="2004"/>
        <enumeration value="2005"/>
        <enumeration value="2100"/>
        <enumeration value="2101"/>
        <enumeration value="2102"/>
        <enumeration value="2103"/>
        <enumeration value="2104"/>
        <enumeration value="2105"/>
```

```
        <enumeration value="2106"/>
        <enumeration value="2200"/>
        <enumeration value="2201"/>
        <enumeration value="2202"/>
        <enumeration value="2300"/>
        <enumeration value="2301"/>
        <enumeration value="2302"/>
        <enumeration value="2303"/>
        <enumeration value="2304"/>
        <enumeration value="2305"/>
        <enumeration value="2306"/>
        <enumeration value="2307"/>
        <enumeration value="2308"/>
        <enumeration value="2400"/>
        <enumeration value="2500"/>
        <enumeration value="2501"/>
        <enumeration value="2502"/>
      </restriction>
    </simpleType>

<!-- Fin du schéma. -->
  </schema>
 FIN
```

4.2 Schéma de structure partagé

DÉBUT

```
  <?xml version="1.0" encoding="UTF-8"?>

  <schema targetNamespace="urn:ietf:params:xml:ns:eppcom-1.0"
        xmlns:eppcom="urn:ietf:params:xml:ns:eppcom-1.0"
        xmlns="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">

    <annotation>
      <documentation>
        Extensible Provisioning Protocol v1.0
        shared structures schema.
      </documentation>
    </annotation>
```

```
<!-- Types d'informations d'autorisation d'objet. -->
    <complexType name="pwAuthInfoType">
      <simpleContent>
        <extension base="normalizedString">
          <attribute name="roid" type="eppcom:roidType"/>
        </extension>
      </simpleContent>
    </complexType>

    <complexType name="extAuthInfoType">
      <sequence>
        <any namespace="##other"/>
      </sequence>
    </complexType>

<!-- Types de réponse <check>. -->
    <complexType name="reasonType">
      <simpleContent>
        <extension base="eppcom:reasonBaseType">
          <attribute name="lang" type="language"/>
        </extension>
      </simpleContent>
    </complexType>

    <simpleType name="reasonBaseType">
      <restriction base="token">
        <minLength value="1"/>
        <maxLength value="32"/>
      </restriction>
    </simpleType>

<!-- Type abstrait d'identifiant de client et d'objet. -->
    <simpleType name="clIDType">
      <restriction base="token">
        <minLength value="3"/>
        <maxLength value="16"/>
      </restriction>
    </simpleType>

<!-- Type d'étiquette DNS. -->

    <simpleType name="labelType">
      <restriction base="token">
        <minLength value="1"/>
        <maxLength value="255"/>
      </restriction>
    </simpleType>

  <!--   Type de jeton non vide.   -->
    <simpleType name="minTokenType">
      <restriction base="token">
        <minLength value="1"/>
      </restriction>
    </simpleType>

<!-- Type d'identifiant d'objet de répertoire. -->
    <simpleType name="roidType">
      <restriction base="token">
        <pattern value="(\w|_){1,80}-\w{1,8}"/>
      </restriction>
    </simpleType>
```

```
<!-- Identifiants d'état de transfert. -->

    <simpleType name="trStatusType">
      <restriction base="token">
        <enumeration value="clientApproved"/>
        <enumeration value="clientCancelled"/>
        <enumeration value="clientRejected"/>
        <enumeration value="pending"/>
        <enumeration value="serverApproved"/>
        <enumeration value="serverCancelled"/>
      </restriction>
    </simpleType>

<!-- Fin de schéma. -->
  </schema>
FIN
```

5. Considérations d'internationalisation

EPP est représenté en XML, qui fournit la prise en charge native des informations de codage en utilisant le jeu de caractères Unicode et ses représentations plus compactes incluant UTF-8. Les processeurs conformes à XML reconnaissent UTF-8 et UTF-16. Bien que XML comporte des dispositions pour identifier et utiliser d'autres codages de caractères par l'utilisation d'un attribut "encoding" dans une déclaration <?xml?>, l'utilisation de UTF-8 est RECOMMANDÉE dans les environnements où existent des incompatibilités de prise en charge d'analyseur de codage.

EPP comporte une disposition pour retourner des messages lisibles par l'homme avec tout code de résultat. Le présent document décrit les codes de résultat en anglais, mais le texte réel retourné avec un résultat PEUT être fourni dans un langage négocié lorsque une session est établie. Les langages autres que l'anglais DOIVENT être notés par la spécification d'un attribut "lang" pour chaque message. Les valeurs valides pour l'attribut "lang" et les éléments de négociation de "lang" sont décrits dans la [RFC4646].

Toutes les valeurs de date et heure présentées via EPP DOIVENT être exprimées en temps universel coordonné en utilisant le calendrier grégorien. Le schéma XML permet l'utilisation d'identifiants de zone horaire pour indiquer les décalages au méridien zéro, mais cette option NE DOIT PAS être utilisée avec EPP. La forme étendue de date-heure utilisant les caractères majuscules "T" et "Z" définie dans [W3C.REC-xmlschema-2-20041028] DOIT être utilisée pour représenter les valeurs de date-heure, car le schéma XML ne prend pas en charge les formes tronquées de date-heure ou les caractères "T" et "Z" en minuscules.

6. Considérations relatives à l'IANA

Le présent document utilise des URN pour décrire les espaces de noms XML et les schémas XML conformes à un mécanisme d'enregistrement décrit dans la [RFC3688]. Quatre allocations d'URI ont été enregistrées par l'IANA.

Demande d'enregistrement pour l'espace de noms EPP :
URI : urn:ietf:params:xml:ns:epp-1.0
Contact : voir la section "Adresse de l'auteur" du présent document.
XML : aucun. Les URI d'espace de noms ne représentent pas une spécification XML.

Demande d'enregistrement pour le schéma XML EPP :
URI : urn:ietf:params:xml:schema:epp-1.0
Contact : voir la section "Adresse de l'auteur" du présent document.
XML : voir la section "Schéma de base" du présent document.

Demande d'enregistrement pour l'espace de noms de structure partagée EPP :
URI : urn:ietf:params:xml:ns:eppcom-1.0
Contact : voir la section "Adresse de l'auteur" du présent document.
XML : aucun. Les URI d'espace de noms ne représentent pas une spécification XML.

Demande d'enregistrement pour le schéma XML de structure partagée EPP :
URI : urn:ietf:params:xml:schema:eppcom-1.0
Contact : voir la section "Adresse de l'auteur" du présent document.
XML : voir la section "Schéma de structure partagée" du présent document.

Un gabarit d'enregistrement de type de support MIME figure à l'Appendice B.

7. Considérations pour la sécurité

EPP fournit seulement de simples services d'authentification de client. Une attaque passive est suffisante pour récupérer des identifiants et mots de passe de clients, permettant une falsification triviale de commandes. La protection contre les attaques les plus courantes et des services de sécurité plus robustes DOIVENT être fournis par les autres couches de protocole. Précisément, les instances de EPP DOIVENT être protégées en utilisant un mécanisme de transport ou un protocole d'application qui assure l'intégrité, la confidentialité, et une authentification mutuelle forte entre client et serveur.

EPP utilise une variante du mécanisme SASL PLAIN décrit dans la [RFC4616] pour fournir un service d'authentification simple de couche application qui augmente ou complète les services d'authentification et d'identification qui peuvent être disponibles aux autres couches de protocole. Là où le mécanisme SASL PLAIN spécifie la fourniture d'un identifiant d'autorisation, d'un identifiant d'authentification, et d'un mot de passe comme une seule chaîne séparée par des caractères ASCII NUL, EPP spécifie l'utilisation d'un identifiant combiné d'autorisation et d'authentification et un mot de passe fournis comme éléments XML distincts.

Des tentatives répétées de deviner un mot de passe peuvent être découragées en limitant le nombre de tentatives de <login> sur une connexion ouverte. Un serveur PEUT clore une connexion ouverte si plusieurs tentatives de <login> sont faites avec un identifiant de client invalide, un mot de passe invalide, ou à la fois un identifiant de client et un mot de passe invalides.

EPP utilise les informations d'authentification associées aux objets pour confirmer l'autorité de transfert d'objet. Les informations d'authentification échangées entre les clients EPP et des entités tierces DOIVENT être échangées en utilisant une facilité qui assure des services de confidentialité et d'intégrité pour protéger contre la divulgation non autorisée et la modification pendant le transit.

Les instances EPP DEVRAIENT être protégées en utilisant un mécanisme de transport ou un protocole d'application qui assure la protection contre la répétition. EPP fournit une certaine protection contre les attaques en répétition par l'idempotence des commandes et l'identification de transaction initiée par le client. Des répétitions consécutives de commandes ne vont en aucune façon changer l'état d'un objet. Il y a cependant des chances qu'il y ait des conséquences inattendues ou malveillantes si une commande est répétée après que des commandes intermédiaires ont changé l'état de l'objet et si un identifiant de client n'est pas utilisé pour détecter les répétitions. Par exemple, une commande <create> répétée qui suit une commande <delete> pourrait réussir si il n'y a pas de facilités supplémentaires pour empêcher ou détecter la répétition.

Comme décrit à la Section 2, EPP comporte des dispositifs qui permettent une revue hors ligne des commandes de transformation avant que l'action demandée soit réellement achevée. Le serveur est obligé de notifier au client que le traitement hors ligne de l'action a été achevé. Les notifications peuvent être envoyées en utilisant un mécanisme hors bande qui n'est pas protégé par le mécanisme utilisé pour assurer la sécurité du transport EPP. Les notifications envoyées sans les services de sécurité du transport de EPP devraient être protégées en utilisant un autre mécanisme qui fournisse un niveau approprié de protection de la notification.

8. Remerciements

La RFC 3730 est un produit du groupe de travail PROVREG, qui a suggéré des améliorations et a fourni de nombreux commentaires précieux. L'auteur souhaite remercier de leurs efforts les présidents du groupe de travail Edward Lewis et Jaap Akkerhuis pour leurs contributions aux processus et à la rédaction. La RFC 4930 et le présent document sont des soumissions individuelles, fondées sur le travail fait dans la RFC 3730.

Les suggestions spécifiques qui ont été incorporées dans le présent document ont été fournies par Chris Bason, Eric Brunner-Williams, Jordyn Buchanan, Roger Castillo Cortazar, Dave Crocker, Ayesha Damaraju, Sheer El-Showk, Patrik Faltstrom, James Gould, John Immordino, Dan Kohn, Hong Liu, Klaus Malorny, Dan Manley, Michael Mealling, Patrick Mevzek, Andrew Newton, Budi Rahardjo, Asbjorn Steira, Rick Wesson, et Jay Westerdal.

9. Références

9.1 Références normatives

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

[RFC2277] H. Alvestrand, "Politique de l'IETF en matière de jeux de caractères et de langages", BCP 18, janvier 1998.

[RFC2914] S. Floyd, "Principes du contrôle d'encombrement", BCP 41, septembre 2000.

[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.

[RFC4646] A. Phillips, M. Davis, "Étiquettes d'identification des langues", BCP0047 septembre 2006. *(Remplacée par* RFC5646*)*

[W3C.REC-xml-20040204] Sperberg-McQueen, C., Maler, E., Yergeau, F., Paoli, J., and T. Bray, "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 >.

[W3C.REC-xmlschema-1-20041028] Maloney, M., Thompson, H., 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 >.

[W3C.REC-xmlschema-2-20041028] Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, octobre 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028 >.

9.2 Références pour information

[RFC0793] J. Postel (éd.), "Protocole de commande de transmission – Spécification du protocole du programme Internet DARPA", STD 7, septembre 1981.

[RFC2781] P. Hoffman et F. Yergeau, "UTF-16, un codage de la norme ISO 10646", février 2000.

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

[RFC3080] M. Rose, "Cœur du protocole extensible d'échange de blocs (BEEP)", mars 2001. *(P.S.)*

[RFC3375] S. Hollenbeck, "Exigences pour le protocole générique de registre-registraire", septembre 2002. *(Information)*

[RFC4616] K. Zeilenga, éd., "Mécanisme PLAIN de l'authentification simple et couche de sécurité (SASL)", août 2006. *(P.S.)*

[RFC4930] S. Hollenbeck, "Protocole d'approvisionnement extensible (EPP)", mai 2007. *(D.S., remplacée par la* RFC5730*)*

[RFC4960] R. Stewart, éd., "Protocole de transmission de commandes de flux", septembre 2007. (*Remplace* RFC2960, RFC3309) *(P.S.)*

[RFC5321] J. Klensin, "Protocole simple de transfert de messagerie", octobre 2008. (*Remplace* RFC2821) (*MàJ* RFC1123) *(D.S.)*

[W3C.REC-P3P-20020416] Marchiori, M., "The Platform for Privacy Preferences 1.0 (P3P1.0) Specification", World Wide Web Consortium Recommendation REC-P3P-20020416, avril 2002, http://www.w3.org/TR/2002/REC-P3P-20020416>.

Appendice A Gabarit de transposition d'objet

Le présent appendice décrit une présentation recommandée pour documenter la transposition en EPP d'un objet. Les documents qui décrivent des transpositions d'objet EPP DEVRAIENT décrire la transposition dans un format similaire à celui utilisé ici. Des sections supplémentaires sont nécessaires si la transposition d'objet est écrite dans un format de projet Internet ou de RFC.

1. Introduction
Fournit l'introduction qui décrit l'objet et donne une vue d'ensemble de la transposition en EPP.

2. Attributs d'objet
Décrit les attributs associés à l'objet, incluant des références appropriées aux spécifications de syntaxe. Des exemples d'attributs d'objet incluent un nom ou identifiant et les dates associées aux événements de modification.

3. Transposition de commande EPP
3.1 Commandes d'interrogation EPP
3.1.1 Commande EPP <check>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande EPP <check>. Inclut à la fois des échantillons de commandes et de réponses.

3.1.2 Commande EPP <info>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande EPP <info>. Inclut à la fois des échantillons de commandes et de réponses.

3.1.3 Commande EPP <poll>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande EPP <poll>. Inclut à la fois des échantillons de commandes et de réponses.

3.1.4 Commande EPP <transfer>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande d'interrogation EPP <transfer>. Inclut à la fois des échantillons de commandes et de réponses.

3.2 Commandes de transformation EPP
3.2.1 Commande EPP <create>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande EPP <create>. Inclut à la fois des échantillons de commandes et de réponses. Décrit l'état de l'objet par rapport au temps, incluant le comportement attendu du client et du serveur si une période de validité est utilisée.

3.2.2 Commande EPP <delete>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande EPP <delete>. Inclut à la fois des échantillons de commandes et de réponses.

3.2.3 Commande EPP <renew>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande EPP <renew>. Inclut à la fois des échantillons de commandes et de réponses.

3.2.4 Commande EPP <transfer>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande EPP <transfer>. Inclut à la fois des échantillons de commandes et de réponses.

3.2.4 Commande EPP <update>
Décrit les transpositions spécifiques de l'objet requises pour mettre en œuvre la commande EPP <update>. Inclut à la fois des échantillons de commandes et de réponses.

4. Syntaxe formelle
Fournit le schéma XML pour la transposition de l'objet. Un DTD XML NE DOIT PAS être utilisé, car les DTD ne fournissent pas une prise en charge suffisante pour les espaces de noms XML et une forte caractérisation des données.

Appendice B Enregistrement de type de support : application/epp+xml

Nom du type de support MIME : application

Nom du sous type MIME : epp+xml
Paramètres exigés : aucun
Paramètres facultatifs : les mêmes que le paramètre de jeu de caractère de application/xml spécifié dans la [RFC3023].
Considérations de codage : les mêmes que les considérations de codage de application/xml spécifié dans la [RFC3023].
Considérations de sécurité : ce type a toutes les considérations de sécurité décrites dans la [RFC3023] plus les considérations spécifiées dans la section Considérations pour la sécurité du présent document.
Considérations d'interopérabilité : l'interopérabilité de XML entre les clients et serveurs d'auteurs et versions distribuées WWW (WebDAV) a été prouvée, et pour l'importation et l'exportation à partir de plusieurs outils XML de collecte de noms d'auteurs. Pour une interopérabilité maximale, des processeurs de validation sont recommandés. Bien que des processeurs non valideurs puissent être plus efficaces, ils ne sont pas exigés pour traiter toutes les caractéristiques de XML. Pour plus d'informations, voir le paragraphe 2.9, "Déclaration de document autonome", et la Section 5, "Conformité", de [W3C.REC-xml-20040204].

Spécification publiée : ce document.

Applications qui utilisent ce type de support : EPP est neutre à l'égard de l'appareil, de la plate-forme, et du fabricant et est pris en charge par plusieurs fournisseurs de services.

Informations supplémentaires : si ils sont utilisés, les numéros magiques, les identifiants de fragments, les URI de base, et l'utilisation du BOM devraient être comme spécifié dans la [RFC3023].

Numéros magiques : aucun
Extension de fichier : xml
Code de type de fichier Macintosh : "TEXT"
Adresse personnelle & de messagerie électronique pour plus d'informations : voir l'adresse de l'auteur.
Utilisation prévue : COMMUNE
Auteur/contrôleur des changements : IETF

Appendice C Changements par rapport à la RFC 4930

1. On a changé "Le présent document rend obsolète la RFC 3730" en "Le présent document rend obsolète la RFC 4930".

2. On a remplacé la référence à la RFC 2595 par la référence à la RFC 4616.

3. On a remplacé la référence à la RFC 2821 par la référence à la RFC 5321.

4. On a remplacé la référence à la RFC 2960 par la référence à la RFC 4960.

5. On a remplacé la référence à la RFC 3066 par la référence à la RFC 4646.

6. On a remplacé la référence à la RFC 3730 par la référence à la RFC 4930.

7. Ajout de "Un client de protocole qui est autorisé à gérer un objet existant est décrit comme un client "garant" partout dans le présent document" au paragraphe 1.1.

8. On a changé "Cette action DOIT être ouverte à tous les clients autorisés" en "Cette commande DOIT être disponible à tous les clients" dans les descriptions des commandes <login> et <logout>.

9. On a changé "Des codes de résultat spécifiques sont énumérés dans le tableau ci-dessous" en "La liste complète des codes de résultat valides figure ci-dessous et dans le schéma normatif" à la Section 3.

10. Ajout d'un nouveau paragraphe à la Section 7 pour donner des lignes directrices sur le besoin de protéger les notices de transaction hors ligne.

11. Ajout d'une référence à l'Appendice B dans la section des considérations relatives à l'IANA.

12. Ajout du texte de licence BSD à la section de schéma XML.

Adresse de l’auteur

Scott Hollenbeck
VeriSign, Inc.
21345 Ridgetop Circle
Dulles, VA 20166-6503
USA
mél : shollenbeck@verisign.com