Groupe de travail Réseau
Request for Comments : 4662
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

A. B. Roach & B. Campbell, Estacado Systems
J. Rosenberg, Cisco Systems
août 2006

Extension de notification d'événement du protocole d'initialisation de session (SIP) pour les listes de ressources

Résumé
Le présent document présente une extension au mécanisme de notification d'événement spécifique du protocole d'initialisation de session (SIP, *Session Initiation Protocol*) pour l'abonnement à une liste homogène de ressources. Au lieu d'envoyer un SUBSCRIBE pour chaque ressource individuellement, l'abonné peut souscrire à une liste entière et recevoir ensuite des notifications quand l'état d'une des ressources de la liste change.

Table des matières

1. Introduction

Le mécanisme de notification d'événement spécifique de SIP [RFC3265] permet à un utilisateur (l'abonné) de demander à recevoir une notification des changements de l'état d'une ressource particulière. Ceci se fait par la génération par l'abonné d'une demande SUBSCRIBE pour la ressource, demande qui est traitée par un notificateur qui représente la ressource.

Dans de nombreux cas, un abonné a une liste des ressources qui l'intéressent. Sans un mécanisme d'agrégation, cela va exiger que l'abonné génère une demande SUBSCRIBE pour chaque ressource sur laquelle il veut des informations. Pour les environnements dans lesquels la bande passante est limitée, comme les réseaux sans fil, l'abonnement à chaque ressource individuelle est problématique. Certains des problèmes spécifiques sont :

- le faire génère un trafic substantiel de messages, sous la forme de demandes SUBSCRIBE initiales pour chaque ressource et le rafraîchissement de chaque abonnement individuel ;
- le notificateur peut insister pour diminuer les intervalles de rafraîchissement, afin d'éviter un état d'abonnement de longue durée. Cela signifie que l'abonné peut avoir besoin de générer des rafraîchissements de SUBSCRIBE plus rapidement qu'il le voudrait ou qu'il a la capacité de le faire ;
- le notificateur peut générer des demandes NOTIFY plus rapidement que ne le désire l'abonné, causant un trafic de NOTIFY de plus grand volume que désiré par l'abonné.

Pour résoudre ces problèmes, la présente spécification définit une extension à la [RFC3265] qui permet de demander et transporter des notifications pour des listes de ressources. Une liste de ressources est identifiée par un URI, et elle représente une liste de zéro, un ou plusieurs URI. Chacun de ces URI est un identifiant pour une ressource individuelle pour laquelle l'abonné veut recevoir les informations. Dans de nombreux cas, l'URI utilisé pour identifier la liste de ressources va être un URI SIP [RFC3261] ; cependant, l'utilisation d'autres schémas (comme par exemple [RFC3859]) est aussi prévue.

Le notificateur pour la liste est appelé un "serveur de liste de ressources" (RLS, *Resource List Server)*. Afin de déterminer l'état de la liste entière, le RLS va agir comme si il avait généré un abonnement à chaque ressource de la liste.

La liste de ressources n'est pas restreinte à être dans le domaine de l'abonné. De même, les ressources de la liste ne sont pas contraintes à être dans le domaine du serveur de liste de ressources.

2. Terminologie

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDÉ", "PEUT", et "FACULTATIF" sont à interpréter comme décrit dans le BCP 14, [RFC 2119] et indiquent les niveaux d'exigence pour les mises en œuvre conformes.

Les termes suivants sont utilisés dans tout le reste de ce document.

Abonnement spécialisé : tout abonnement (SIP ou autre) que crée un RLS pour apprendre l'état d'une ressource. Un RLS va créer des abonnements spécialisés pour apprendre l'état d'une ressource sur laquelle le RLS n'est pas une autorité. Pour les abonnements spécialisés, les RLS agissent comme un abonné.

Abonnement de listes : abonnement à une liste de ressources. Dans les abonnements de listes, les RLS agissent comme notificateur.

Ressource : toute entité logique qui a un ou des états auxquels on peut s'abonner. Les ressources sont identifiées par des URI.

Liste de ressources : liste de zéro, une ou plusieurs ressources dont les états individuels peuvent être souscrits avec un seul abonnement.

RLMI *(Resource List Meta-Information)* : méta informations de liste de ressources ; c'est un document qui décrit l'état des abonnements virtuels associés à un abonnement de liste.

RLS *(Resource List Server)* : les serveurs de liste d'abonnement acceptent des abonnements aux listes de ressource et envoient des notifications pour que les abonnés mettent à jour l'état des ressources d'une liste de ressources.

Abonnement virtuel : c'est une construction logique dans un RLS qui représente des abonnements aux ressources d'une liste de ressources. Pour chaque abonnement de liste qu'il sert, un RLS crée au moins un abonnement virtuel pour chaque ressource de la liste de ressources qui fait l'objet de l'abonnement. Dans certains cas, comme quand le RLS n'est pas l'autorité pour l'état de la ressource, cet abonnement virtuel va être associé à un abonnement spécialisé. Dans d'autres cas, comme quand le RLS est l'autorité pour l'état de la ressource, l'abonnement virtuel ne va pas avoir d'abonnement spécialisé correspondant.

3. Généralités sur le fonctionnement

Cette Section donne une vue d'ensemble du mode de fonctionnement normal de cette extension. Elle n'est pas normative.

Quand des utilisateurs souhaitent s'abonner aux ressources d'une liste de ressources, ils peuvent utiliser les mécanismes décrits dans la présente spécification. La première étape est la création d'une liste de ressources. Cette liste de ressources est représentée par un URI SIP. La liste contient un ensemble d'URI, dont chacun représente une ressource pour laquelle l'abonné veut recevoir les informations. La liste de ressources peut exister dans tout domaine. La liste pourrait être manipulée par une page de la Toile, un système de réponse vocale, ou par un autre protocole. Les moyens spécifiques par lesquels la liste est créée et maintenue sortent du domaine d'application de la présente spécification.

Pour apprendre l'état d'une ressource de l'ensemble d'éléments de la liste, l'utilisateur envoie une seule demande SUBSCRIBE ciblée par l'URI de la liste. Elle va être acheminée à un RLS pour cet URI. Le RLS agit comme notificateur, authentifie l'abonné, et accepte l'abonnement.

Le RLS peut avoir des informations directes sur tout ou partie des ressources spécifiées par la liste. Si il n'en a pas, il pourrait s'abonner à toute ressource non locale spécifiée par les ressources de la liste.

Noter que les abonnements à des ressources non locales peuvent être ou non des abonnements SIP ; tout mécanisme pour déterminer de telles informations peut être employé. Le présent document utilise le terme de "abonnement spécialisé" pour se référer à de tels abonnements, sans considérer si SIP est utilisé pour l'établir et le servir.

Lorsque l'état des ressources de la liste change, le RLS génère des notifications aux abonnés à la liste. Le RLS peut, à sa discrétion, mettre en mémoire tampon les notifications de changement de ressources et envoyer les informations de ressources par lots, plutôt que individuellement. Cela permet au RLS de limiter le débit pour l'abonné.

Les notifications de liste contiennent un corps de type multipart/related. La section racine du contenu multipart/related est un document XML qui fournit des méta informations sur chaque ressource présente dans la liste. Les Sections suivantes contiennent les informations d'état réelles pour chaque ressource.

4. Fonctionnement des abonnements de liste

L'extension Liste d'événements agit, à beaucoup d'égards, comme un paquetage de gabarit d'événement. En particulier, tout abonnement de liste doit être homogène par rapport au paquetage d'événement sous-jacent. En d'autres termes, un seul abonnement de liste peut s'appliquer seulement à un paquetage d'événement pour toutes les ressources de la liste de ressources.

Noter qu'il est parfaitement valide pour un RLS de permettre que plusieurs abonnements à la même liste utilisent des paquetages d'événement différents.

La différence clé entre un abonnement de liste et les gabarits en général est que la prise en charge des abonnements de liste indique la prise en charge d'une incorporation arbitraire d'abonnements de liste. En d'autres termes, les éléments dans la liste peuvent être des éléments atomiques, ou ils peuvent être eux-mêmes des listes.

Cela a pour conséquence qu'un abonnement à un URI qui représente une liste résulte en fait en plusieurs abonnements virtuels à une arborescence de ressources. Les nœuds d'extrémité de cette arborescence sont les abonnements virtuels du type d'événement donné dans le champ d'en-tête "Événement" ; tous les autres nœuds dans l'arborescence sont des abonnements de liste qui sont servis comme décrit dans cette Section et ses paragraphes.

On se souviendra que ces abonnements virtuels ne sont pas littéralement des abonnements SIP (bien qu'ils puissent résulter en abonnements SIP, selon la mise en œuvre de RLS).

4.1 Négociation de la prise en charge des listes de ressources

La présente spécification utilise le mécanisme d'étiquette d'option SIP pour négocier la prise en charge de l'extension définie ici. On se reportera à la [RFC3261] pour la description normative du traitement des champs d'en-tête "Supported" et "Require" et le code de réponse 421 (Extension exigée).

Voici une description non normative des implications de l'utilisation des étiquettes d'option.
Tout client qui prend en charge l'extension Liste d'événements va inclure une étiquette d'option de "eventlist" dans un champ d'en-tête "Supported" de chaque message SUBSCRIBE pour un abonnement pour lequel il veut traiter une liste. Si l'abonnement est fait à un URI qui représente une liste, le RLS va inclure "eventlist" dans un champ d'en-tête "Require" de la réponse au SUBSCRIBE, et dans tous les messages NOTIFY au sein de cet abonnement.

L'utilisation de "Require: eventlist" dans les messages NOTIFY est appliquée par le notificateur pour satisfaire l'exigence de la RFC 3261 qu'un UAC DOIT insérer un champ d'en-tête Require dans une demande si l'UAC souhaite insister pour qu'un UAS comprenne une extension afin de traiter la demande. Parce que le NOTIFY ne serait pas utilisable sans appliquer l'option eventlist, le notificateur est obligé de l'inclure.

Inclure "eventlist" dans un champ d'en-tête "Require" dans un message SUBSCRIBE ne sert à rien d'autre que casser l'interopérabilité dans certains cas, et est par conséquent NON RECOMMANDÉ.

L'envoi de "Supported: eventlist" dans un message NOTIFY n'a pas de signification et est stupide. Les mises en œuvre NE DEVRAIENT PAS inclure "Supported: eventlist" dans toute demande sauf pour SUBSCRIBE.

Il n'y a rien dans un URI SIP qui indique si il représente une liste de ressources ou une seule ressource. Donc, si un abonné envoie une demande à un URI qui représente une ressource de liste mais si il n'inclut pas de champ d'en-tête Supported faisant la liste des jetons "eventlist", le notificateur va normalement retourner un code de réponse 421 (Extension exigée). La [RFC3261] conseille que les serveurs évitent de retourner un 421 et tentent plutôt de traiter la demande sans l'extension. Cependant, dans ce cas, l'URI représente fondamentallement une ressource de liste, et donc, l'abonnement ne peut pas se poursuivre sans cette extension.

4.2 Durée d'abonnement

Comme le principal avantage du serveur de liste de ressources est de réduire le volume global de messages à un abonné, il est RECOMMANDÉ que la durée de l'abonnement à une liste soit raisonablement long. Par défaut, quand aucune durée n'est spécifiée, le paquetage d'événement sous-jacent est utilisé. Bien sûr, les techniques standard de la [RFC3265] peuvent être utilisées pour augmenter ou réduire cette durée.

4.3 Corps NOTIFY

Une mise en œuvre conforme à la présente spécification DOIT prendre en charge les types MIME multipart/related et application/rlmi+xml. Ces types DOIVENT être inclus dans un en-tête Accept envoyé dans un message SUBSCRIBE, en plus de tous les autres types pris en charge par le client (incluant tous types exigés par le paquetage d'événement utilisé).

4.4 Traitement par le RLS des demandes SUBSCRIBE

Une fois que l'abonné est authentifié, le RLS effectue l'autorisation selon sa politique locale. Souvent, chaque liste de ressources est associée à un utilisateur particulirer (celui qui l'a créée et gère l'ensemble d'éléments quelle contient) et seul cet utilisateur va être autorisé à s'y abonner. Bien sûr, ce mode de fonctionnement n'est pas inhérent à l'utilisation des listes de ressources, et un RLS peut utiliser toute politique d'autorisation de son choix.

4.5 Génération par le RLS des demandes NOTIFY

La présente spécification laisse à la discrétion de la mise en œuvre le choix de quand et comment générer les demandes NOTIFY. Un des éléments de différentiation entre diverses mises en œuvre de RLS est le moyen par lequel elles agrègent, limitent en débit, ou optimisent la façon dont les notifications sont générées. Comme comportement de base, le RLS PEUT générer un NOTIFY à l'abonné RLS chaque fois que change l'état d'une ressource sur la liste.

Il est important de comprendre que tout abonnement est un abonnement soit à une seule ressource soit à une liste de ressources. Cette nature (seule ressource contre liste de ressources) ne peut pas changer pendant la durée d'un seul abonnement. En particulier, cela signifie que les RLS NE DOIVENT PAS envoyer de messages NOTIFY qui ne contiennent pas de RLMI pour

un abonnement si ils ont précédemment envoyé des messages NOTIFY contenant des RLMI dans cet abonnement. De même, les RLS NE DOIVENT PAS envoyer de messages NOTIFY contenant des RLMI pour un abonnement si ils ont précédemment envoyé des messages NOTIFY qui n'en contenaient pas dans cet abonnement.

Les représentations de liste contiennent nécessairement des documents RLMI pour deux raisons. La plus importante est qu'ils identifient la ressource à laquelle correspond l'état d'événement. De nombreuses syntaxes d'état n'identifient pas pleinement la ressource à laquelle l'état s'applique, ou elles peuvent identifier la ressource d'une façon différente de celle dont elle est représentée dans la liste ; par exemple, les documents PIDF peuvent contenir des URI de ressource qui ne sont pas identiques à l'URI utilisé pour la restituer. De plus, les documents RLMI servent à distinguer les multiples instances d'une seule ressource.

Voir à la Section 5 une définition détaillée de la syntaxe utilisée pour convoyer l'état des listes de ressources. Pour les besoins de la discussion qui suit, il est important de savoir que la liste globale contient zéro, une ou plusieurs ressources, et que les ressources contiennent zéro, une ou plusieurs instances. Chaque instance a un état associé (imminant, actif, ou en clôture) représentant l'état de l'abonnement virtuel.

Les notifications contiennent un document multi parties, dont la première contient toujours des méta informations sur la liste (par exemple, les membres, l'état de l'abonnement virtuel à la ressource). Les parties restantes sont utilisées pour porter l'état réel des ressources mentionnées dans les méta informations.

L'attribut "état" de chaque instance d'une ressource dans les méta informations est réglé en accord avec l'état de l'abonnement virtuel. La signification de l'attribut "état" est décrite dans la [RFC3265].

Si une instance d'une ressource a été précédemment rapportée à l'abonné mais n'est plus disponible (c'est-à-dire, l'abonnement virtuel à cette instance a été terminé) le serveur de liste de ressources DEVRAIT inclure cette instance de ressource dans les méta informations dans le premier message NOTIFY envoyé à l'abonné qui suit l'indisponibilité de l'instance. Le RLS PEUT continuer de faire cela pour les notifications futures.

Quand il envoie des informations pour une instance de ressource terminée, le RLS indique un état de "terminé" et une valeur de raison appropriée. Les valeurs de raison valides et leur signification sont décrites dans la [RFC3265]. Si le RLS veut tenter de récupérer à nouveau l'état de ressource à l'avenir (par exemple, quand la raison dans les méta informations est "à l'épreuve") alors l'instance de ressource DEVRAIT rester dans les méta informations tant que l'état de l'instance reste disponible, ou jusqu'à ce que le RLS arrête de rendre cet état disponible.

Quand le premier message SUBSCRIBE pour un abonnement particulier est reçu par un RLS, le RLS va souvent ne pas connaître les informations d'état pour toutes les ressources spécifiées par la liste de ressources. Pour toute ressource pour laquelle les informations d'état ne sont pas connues, l'attribut "uri" correspondant va être réglé de façon appropriée, et aucun élément <instance> ne va être présent pour la ressource.

Pour une notification initiale, les sections correspondant aux ressources pour lesquelles le RLS a bien un état vont être remplies avec les données appropriées (sous réserve, bien sûr, des décisions de politique locale). Cela va souvent se produire si le serveur de liste de ressources est co-localisé avec le serveur pour une ou plusieurs des ressources spécifiées par la liste.

Les notifications immédiates déclenchées par suite des messages SUBSCRIBE suivants DEVRAIENT inclure un document de RLMI dans lequel l'état plein est indiqué. Le RLS DEVRAIT aussi inclure des informations d'état pour toutes les ressources de la liste pour lesquelles le RLS a un état, sous réserve des restrictions de politique. Cela permet à l'abonné de rafraîchir leur état, et de récupérer des notifications perdues.

4.6 Traitement par l'abonné des demandes NOTIFY

Les notifications pour une liste de ressources peuvent porter des informations sur un sous ensemble de la liste des éléments. Cela signifie qu'un algorithme explicite doit être défini afin de construire un état cohérent et consistant.

Le document XML présent dans la racine du document multipart/related contient un élément <ressource> pour certaines ou toutes les ressources de la liste. Chaque élément <ressource> contient un URI qui identifie de façon univoque la ressource à laquelle cette section correspond. Quand un NOTIFY arrive, il peut contenir un état plein ou partiel (comme indiqué par l'attribut "fullState" de l'élément <list> de niveau supérieur). Si l'état plein est indiqué, le receveur remplace alors tous les états associés à la liste par les entités dans le corps du NOTIFY. Si l'état plein n'est pas indiqué, le receveur du NOTIFY met à jour les informations pour chaque ressource identifiée. Les informations pour toutes les ressources qui ne sont pas identifiées dans le NOTIFY ne sont pas changées, même si elles étaient indiquées dans les messages NOTIFY précédents. Voir plus d'informations au paragraphe 5.6.

Quand l'état plein est indiqué, on note qu'il ne s'applique qu'au document RLMI dans lequel il se produit. En particulier, un des éléments <ressource> dans le document peut à son tour se référer à une autre liste de ressources. Toute sous liste de cette sorte va être détaillée dans son propre document RLMI, qui peut avoir ou non l'état plein indiqué.

On notera de plus que le paquetage d'événements sous-jacent peut avoir ses propres règles pour composer une notification d'état partiel. Lors du traitement des données relatives à ces paquetages, leurs règles s'appliquent (c'est-à-dire, le fait qu'elles aient été rapportées au titre d'une liste ne change pas leur sémantique de notification partielle).

Finalement, on note que par suite de la façon dont fonctionnent les abonnements aux listes de ressources, l'interrogation de l'état d'une ressource peut n'être pas particulièrement utile. Bien que de telles interrogations restituent la liste des ressources, elles ne contiennent pas nécessairement l'état pour certaines ou toutes les ressources de la liste.

4.7 Traitement des demandes fourchées

Le fourchement a peu de sens pour les abonnements aux listes d'événements, car toute l'idée est celle d'une centralisation de la source des notifications. Donc, un abonné à une liste NE DOIT PAS installer plusieurs abonnements quand la demande initiale est fourchée. Si plusieurs réponses sont reçues, elles sont traitées en utilisant les techniques décrites au paragraphe 4.4.9 de la [RFC3265].

4.8 Taux de notifications

Un rôle potentiel du RLS est d'effectuer les limitations de débit au nom de l'abonné. À ce titre, la présente spécification ne rend pas obligatoire de limitation de débit particulière, et laisse plutôt cela à la discrétion de la mise en œuvre.

5. Utilisation de multipart/related pour porter l'état agrégé

Pour porter l'état de plusieurs ressources, l'extension de liste utilise le type MIME "multipart/related". La syntaxe de multipart/related est définie dans "Type de contenu MIME Multiparti/relatif" [RFC2387].

5.1 Syntaxe XML

Le document racine du corps multipart/related DOIT être un document de méta informations de liste de ressources (RLMI, *Resource List Meta-Information*). Il est du type "application/rlmi+xml". Ce document contient les méta-informations pour les ressources contenues dans la notification. Le schéma pour ce document XML est donné ci-dessous.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:rlmi"
        elementFormDefault="qualified"
        xmlns="urn:ietf:params:xml:ns:rlmi"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
        schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:element name="list">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name" minOccurs="0"
                maxOccurs="unbounded" />
        <xs:element ref="ressource" minOccurs="0"
                maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="uri" type="xs:anyURI" use="required" />
      <xs:attribute name="version" type="xs:unsignedInt"
                use="required" />
      <xs:attribute name="fullState" type="xs:boolean"
                use="required" />
      <xs:attribute name="cid" type="xs:string" use="optional" />
      <xs:anyAttribute processContents="lax" />
```

```
    </xs:complexType>
  </xs:element>
  <xs:element name="ressource">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name" minOccurs="0"
                    maxOccurs="unbounded" />
        <xs:element ref="instance" minOccurs="0"
                    maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="uri" type="xs:anyURI" use="required" />
      <xs:anyAttribute processContents="lax" />
    </xs:complexType>
  </xs:element>
  <xs:element name="instance">
    <xs:complexType>
      <xs:sequence>
        <xs:any minOccurs="0" maxOccurs="unbounded"
              processContents="lax" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required" />
      <xs:attribute name="state" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="active" />
            <xs:enumeration value="pending" />
            <xs:enumeration value="terminated" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="reason" type="xs:string"
                    use="optional" />
      <xs:attribute name="cid" type="xs:string" use="optional" />
      <xs:anyAttribute processContents="lax" />
    </xs:complexType>
  </xs:element>
  <xs:element name="name">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute ref="xml:lang" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Voici un exemple de document formaté en utilisant ce schéma :

```
<?xml version="1.0"?>
<list xmlns="urn:ietf:params:xml:ns:rlmi"
      uri="sip:adam-friends@lists.vancouver.example.com"
      version="7" fullState="true">
  <name xml:lang="en">Buddy List</name>
  <name xml:lang="fr">Liste d'amis</name>
  <ressource uri="sip:bob@vancouver.example.com">
    <name>Bob Smith</name>
    <instance id="juwigmtboe" state="active"
              cid="12345.aaa@vancouver.example.com"/>
  </ressource>
  <ressource uri="sip:dave@vancouver.example.com">
```

```
    <name>Dave Jones</name>
    <instance id="hqzsuxtfyq" state="active"
          cid="12345.aab@vancouver.example.com"/>
  </ressource>
  <ressource uri="sip:jim@vancouver.example.com">
    <name>Jim</name>
    <instance id="oflzxqzuvg" state="terminated"
          reason="rejected" />
  </ressource>
  <ressource uri="sip:ed@vancouver.example.com">
    <name>Ed</name>
    <instance id="grqhzsppxb" state="pending"/>
  </ressource>
</list>
```

5.2 Attributs de liste

L'élément <list> présent dans une notification de liste DOIT contenir trois attributs.

Le premier attrbut <list> obligatoire est "uri", qui contient l'uri qui correspond à la liste. Normalement, c'est l'URI auquel la demande SUBSCRIBE a été envoyée.

Le second attribut <list> obligatoire est "version", qui contient un nombre de 0 à 2^32-1. Ce numéro de version DOIT être 0 pour le premier message NOTIFY envoyé dans un abonnement, et DOIT augmenter d'exactement un pour chaque NOTIFY suivant envoyé dans un abonnement.

Le troisième attribut obligatoire est "fullState". L'attribut "fullState" indique si le message NOTIFY contient des informations sur toutes les ressources de la liste. Si c'est le cas, la valeur de l'attribut est "vrai" (ou "1") ; autrement, c'est "faux" (ou "0"). Le premier NOTIFY envoyé dans un abonnement DOIT contenir l'état plein, comme le doit le premier NOTIFY envoyé après la réception d'une demande SUBSCRIBE pour l'abonnement.

Finalement, les éléments <list> PEUVENT contenir un attribut "cid". Si il est présent, l'attribut "cid" identifie une section au sein du corps multipart/related qui contient les informations d'état agrégées pour les ressources contenues dans la liste. La définition de telles informations agrégées sort du domaine d'application de ce document et vont être définies dans chaque paquetage, en tant que de besoin. L'attribut cid est l'identifiant de contenu *(Content-ID)* pour la section correspondante dans le corps multipart.

L'attribut cid DOIT seulement se référer au parties de niveau supérieur du document multipart/related pour lequel le document RLMI dans lequel il apparaît est la racine. Un exemple figure au paragraphe 5.5.

5.3 Attributs de ressource

La liste de ressources contient un élément <ressource> pour chaque ressource rapportée dans la notification. Ces éléments de ressource contiennent des attributs qui identifient les métadonnées associées à chaque ressource.

L'attribut "uri" identifie la ressource à laquelle correspond l'élément <ressource>. Normalement, cela va être un URI SIP qui, si on y est abonné, va retourner l'état de la ressource. Cet attribut DOIT être présent.

5.4 Attributs de nom

Chaque liste et élément de ressource contient zéro, un ou plusieurs éléments de nom. Ces éléments de nom contiennent des descriptions ou noms lisibles par l'homme pour la ressource ou liste de ressources. Le contenu de ces éléments est un peu analogue au "nom d'affichage" présent dans l'élément name-addr de SIP.

Les éléments de nom contiennent facultativement l'attribut standard XML "xml:lang". L'attribut "xml:lang", si il est présent, spécifie le langage d'un nom lisible par l'homme. Si cet attribut est présent, il DOIT contenir une étiquette de langage valide. Les étiquettes de langage sont définies dans la [RFC3066]. L'étiquette de langue aide les applications à déterminer lesquels des potentiellement plusieurs éléments de nom devraient être rendus à l'utilisateur.

5.5 Attributs d'instance

Chaque élément de ressource contient zéro, un ou plusieurs éléments d'instance. Ces éléments d'instance sont utilisés pour représenter un seul notificateur pour la ressource. Pour les paquetages d'événements qui permettent le fourchement, plusieurs abonnements virtuels peuvent exister pour une certaine ressource. Plusieurs abonnements virtuels sont représentés comme plusieurs éléments d'instance dans l'élément de ressource correspondant. Pour les abonnements dans lesquels le fourchement ne se produit pas, au plus une instance va être présente pour une certaine ressource.

L'attribut "id" contient une chaîne opaque utilisée pour identifier de manière univoque l'instance de la ressource. L'attribut "id" n'est unique que dans le contexte d'une ressource. La construction de cette chaîne est une décision de la mise en œuvre. Tout mécanisme pour générer cette chaîne est valide, pour autant que son unicité au sein de la ressource est assurée.

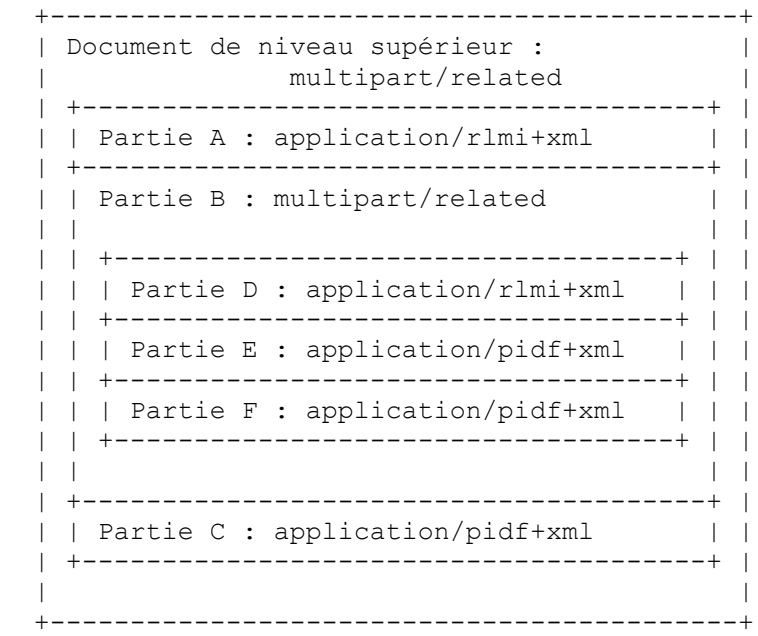
L'attribut "state" contient l'état de l'abonnement pour l'instance identifiée de la ressource. Cet attribut contient une des valeurs "active", "en instance", ou "terminée". La signification de ces valeurs est définie dans le champ d'en-tête "État d'abonnement" dans la [RFC3265].

Si l'attribut "state" indique "terminé", alors un attribut "raison" DOIT aussi être présent. Cet attribut "raison" a les mêmes valeurs et signification que données pour le paramètre "raison" dans le champ d'en-tête "État d'abonnement" dans la [RFC3265]. Noter que l'attribut "raison" est inclus aux fins d'information ; l'abonné à une liste n'est pas supposé effectuer toutes les actions automatiques fondées sur la valeur de raison.

Finallement, l'attribut "cid", qui DOIT être présent si l'attribut "state" est "active", identifie la section au sein du corps multipart/related qui contient l'état réel de la ressource. Cet état est exprimé dans le type de contenu défini par le paquetage d'événement pour l'état porteur. L'attribut cid est le Content-ID pour la section correspondante dans le corps multiparties.

L'attribut cid DOIT se référer seulement aux parties de niveau supérieur du document multipart/related pour lequel le document RLMI dans lequel il apparaît est la racine.

Par exemple, considérons un document multipart/related contenant trois parties ; on va appeler ces parties A, B, et C. La partie A est de type application/rlmi+xml, la partie B est de type multipart/related, et la partie C est de type application/pidf+xml. La partie B est à son tour un document contenant trois parties : D, E, et F. La partie D est du type application/rlmi+xml, et les parties E et F sont du type application/pidf+xml.



Tout attribut "cid" dans le document A doit se référer seulement aux parties B ou C. Se référer aux parties D, E, ou F serait illégal. De même, tous les attributs "cid" dans le document D doivent seulement se référer aux parties E ou F. Se référer à toute autre partie serait illégal. Noter aussi que les durées d'abonnement de tout abonnement spécialisé ne sont de toutes façons pas propagées dans l'état des méta-informations.

5.6 Construction d'un état de ressource cohérent

L'abonné d'une liste de ressources tient un tableau pour chaque liste de ressources. Le tableau contient une rangée pour chaque ressource de la liste de ressources. Chaque rangée est indexée par l'URI pour cette ressource. Cet URI est obtenu de l'attribut "uri" sur chaque élément <ressource>. Le contenu de chaque rangée contient l'état de cette ressource comme porté dans le document de ressource.

Pour les ressources qui fournissent des informations de version (ce qui est rendu obligatoire par la [RFC3265] pour tous les formats qui permettent une notification partielle) chaque rangée contient aussi un numéro de version d'état de ressource. Le numéro de version de la rangée est initialisé avec la version spécifiée dans le premier document reçu, comme défini par le paquetage d'événement correspondant. Cette valeur est utilisée pour comparer les versions des notifications partielles pour une ressource.

Le traitement de la notification de la liste de ressources dépend de si elle contient un état plein ou partiel.

5.6.1 Traitement des notifications d'état plein

Si une notification contient l'état plein, indiqué par l'attribut <list> "fullState" réglé à "vrai", la notification est utilisée pour mettre à jour le tableau. Une vérification est d'abord faite pour s'assurer que l'attribut "version" de l'attribut <list> dans le message reçu est supérieur au numéro de version local. Sinon, le document reçu est éliminé sans autre traitement. Autrement, le contenu du tableau de la liste de ressource est purgé et rempli à partir du contenu du document. Une nouvelle rangée est créée dans le tableau pour chaque élément "ressource".

5.6.2 Traitement des notifications d'état partiel

Si une notification contient un état partiel, indiqué par l'attribut <list> "fullState" réglé à "faux", une vérification est faite pour s'assurer qu'aucune notification de liste n'a été perdue. La valeur du numéro de version local (l'attribut "version" de l'élément <list>) est comparée au numéro de version du nouveau document.

- Si la valeur dans le nouveau document est exactement supérieure de un au numéro de version local, le numéro de version local est augmenté de un, et le document est traité comme décrit ci-dessous.

- Si la version dans le document est supérieure de plus de un au numéro de version local, le numéro de version local est réglé à la valeur dans le nouveau document, et le document est traité comme décrit ci-dessous. L'abonné à la liste DEVRAIT aussi générer une demande de rafraîchissement pour déclencher une notification d'état plein.

- Si la version dans le document est inférieure ou égale à la version locale, le document est éliminé sans autre traitement.

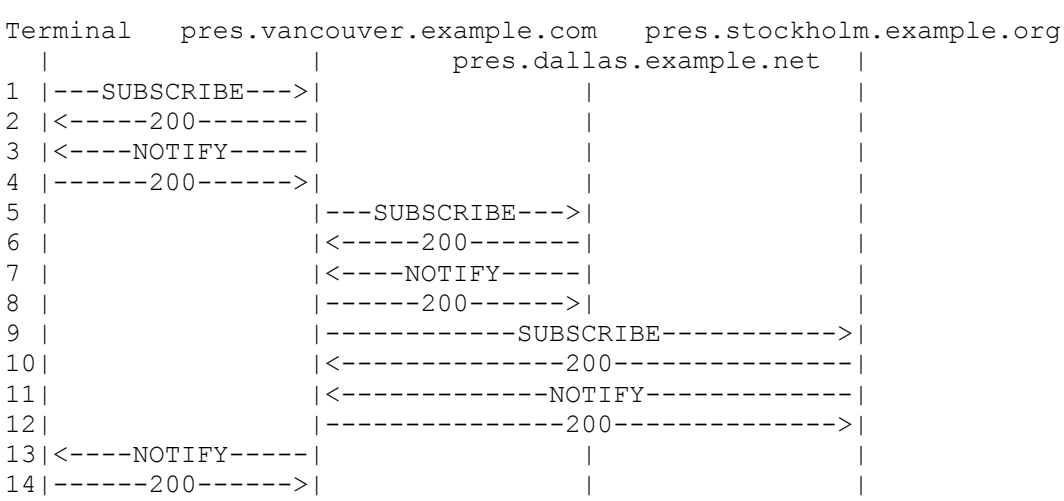
Pour chaque ressource mentionnée dans le document, l'abonné vérifie si une rangée existe pour cette ressource. Cette vérification est faite en comparant la valeur de l'URI de ressource à l'URI associé à la rangée. Si la ressource n'existe pas dans le tableau, une rangée est ajoutée, et son état est réglé aux informations provenant de cet élément "ressource". Si la ressource existe, son état est mis à jour pour être les informations provenant de cet élément "ressource", comme décrit dans la définition du paquetage d'événements. Si une rangée est mise à jour ou créée, de telle façon que son état est maintenant "terminé," cette entrée PEUT être supprimée du tableau à tout moment.

6. Exemple

Cette section donne un exemple de flux d'appels. Elle n'est pas normative. Si un conflit apparaît entre ce flux d'appels et le comportement normatif décrit dans ce document ou tout autre, les descriptions normatives sont à suivre.

Dans cet exemple particulier, on demande un abonnement à une liste de présence incorporée. L'adresse de rattachement de l'abonné est "sip:adam@vancouver.example.com", et le nom de la ressource de liste incorporée à laquelle on s'abonne est appelée "sip:adam-buddies@pres.vancouver.example.com". Le paquetage d'événements sous-jacent est "presence", décrit dans la [RFC3856].

Dans cet exemple, le RLS a des informations à servir à certaines des ressources de la liste, mais doit consulter d'autres serveurs pour restituer les informations pour les autres. La mise en œuvre du RLS dans cet exemple utilise le mécanisme SIP SUBSCRIBE/NOTIFY pour restituer de telles informations.



1. On initie l'abonnement en envoyant un message SUBSCRIBE à notre RLS local. (Il n'y a bien sûr aucune raison pour que le RLS qu'on contacte soit dans notre domaine). Noter qu'on doit annoncer la prise en charge de application/rlmi+xml et multipart/related parce qu'on prend en charge l'extension eventlist, et qu'on doit annoncer application/pidf+xml parce qu'on demande un abonnement à présence.

Terminal -> RLS local

SUBSCRIBE sip:adam-buddies@pres.vancouver.example.com SIP/2.0
Via: SIP/2.0/TCP terminal.vancouver.example.com;
 branch=z9hG4bKwYb6QREiCL
Max-Forwards: 70
To: <sip:adam-buddies@pres.vancouver.example.com>
From: <sip:adam@vancouver.example.com>;tag=ie4hbb8t
Call-ID: cdB34qLToC@terminal.vancouver.example.com
CSeq: 322723822 SUBSCRIBE
Contact: <sip:terminal.vancouver.example.com>
Event: presence
Expires: 7200
Supported: eventlist
Accept: application/pidf+xml
Accept: application/rlmi+xml
Accept: multipart/related
Accept: multipart/signed
Accept: application/pkcs7-mime
Content-Length: 0

2. Le RLS local termine la transaction SUBSCRIBE. Noter que l'authentification et l'autorisation prendraient normalement place à ce point dans le flux d'appels. Ces étapes sont omises pour faire bref.

RLS local -> Terminal

SIP/2.0 200 OK
Via: SIP/2.0/TCP terminal.vancouver.example.com;
 branch=z9hG4bKwYb6QREiCL
To: <sip:adam-buddies@pres.vancouver.example.com>;tag=zpNctbZq
From: <sip:adam@vancouver.example.com>;tag=ie4hbb8t
Call-ID: cdB34qLToC@terminal.vancouver.example.com
CSeq: 322723822 SUBSCRIBE
Contact: <sip:pres.vancouver.example.com>
Expires: 7200
Require: eventlist
Content-Length: 0

3. Comme c'est exigé par la [RFC3265], le RLS envoie un NOTIFY immédiatement à l'acceptation de l'abonnement. Dans cet exemple, on suppose que le RLS local est aussi une autorité pour les informations de présence pour les utilisateurs dans le domaine "vancouver.example.com". Le NOTIFY contient un document RLMI qui décrit la liste complète des amis (les notifications initiales exigent l'état plein) ainsi que les informations de présence pour les utilisateurs qu'il connaît déjà. Noter que, comme le RLS n'a pas encore restitué d'information pour certaines des entrées de la liste, ces éléments <ressource> ne contiennent pas d'élément <instance>.

RLS local -> Terminal

```
NOTIFY sip:terminal.vancouver.example.com SIP/2.0
Via: SIP/2.0/TCP pres.vancouver.example.com;
  branch=z9hG4bKMgRenTETmm
Max-Forwards: 70
From: <sip:adam-buddies@pres.vancouver.example.com>;tag=zpNctbZq
To: <sip:adam@vancouver.example.com>;tag=ie4hbb8t
Call-ID: cdB34qLToC@terminal.vancouver.example.com
CSeq: 997935768 NOTIFY
Contact: <sip:pres.vancouver.example.com>
Event: presence
Subscription-State: active;expires=7200
Require: eventlist
Content-Type: multipart/related;type="application/rlmi+xml";
    start="<nXYxAE@pres.vancouver.example.com>";
    boundary="50UBfW7LSCVLtggUPe5z"
Content-Length: 1560

--50UBfW7LSCVLtggUPe5z
Content-Transfer-Encoding: binary
Content-ID: <nXYxAE@pres.vancouver.example.com>
Content-Type: application/rlmi+xml;charset="UTF-8"

<?xml version="1.0" encoding="UTF-8"?>
<list xmlns="urn:ietf:params:xml:ns:rlmi"
      uri="sip:adam-friends@pres.vancouver.example.com"
      version="1" fullState="true">
  <name xml:lang="en">Buddy List at COM</name>
  <name xml:lang="de">Liste der Freunde an COM</name>
  <ressource uri="sip:bob@vancouver.example.com"">
    <name>Bob Smith</name>
    <instance id="juwigmtboe" state="active"
          cid="bUZBsM@pres.vancouver.example.com"/>
   </ressource>
  <ressource uri="sip:dave@vancouver.example.com">
    <name>Dave Jones</name>
    <instance id="hqzsuxtfyq" state="active"
          cid="ZvSvkz@pres.vancouver.example.com"/>
  </ressource>
  <ressource uri="sip:ed@dallas.example.net">
    <name>Ed at NET</name>
  </ressource>
  <ressource uri="sip:adam-friends@stockholm.example.org">
    <name xml:lang="en">My Friends at ORG</name>
    <name xml:lang="de">Meine Freunde an ORG</name>
  </ressource>
</list>

--50UBfW7LSCVLtggUPe5z
Content-Transfer-Encoding: binary
Content-ID: <bUZBsM@pres.vancouver.example.com>
Content-Type: application/pidf+xml;charset="UTF-8"
```

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
   entity="sip:bob@vancouver.example.com">
  <tuple id="sg89ae">
    <status>
     <basic>open</basic>
   </status>
    <contact priority="1.0">sip:bob@vancouver.example.com</contact>
  </tuple>
</presence>

--50UBfW7LSCVLtggUPe5z
Content-Transfer-Encoding: binary

Content-ID: <ZvSvkz@pres.vancouver.example.com>
Content-Type: application/pidf+xml;charset="UTF-8"

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
   entity="sip:dave@vancouver.example.com">
  <tuple id="slie74">
    <status>
     <basic>closed</basic>
    </status>
  </tuple>
</presence>

--50UBfW7LSCVLtggUPe5z--
```

4. Le terminal termine la transaction.

Terminal -> RLS local

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP pres.vancouver.example.com;
  branch=z9hG4bKMgRenTETmm
From: <sip:adam-buddies@pres.vancouver.example.com>;tag=zpNctbZq
To: <sip:adam@vancouver.example.com>;tag=ie4hbb8t
Call-ID: cdB34qLToC@terminal.vancouver.example.com
CSeq: 997935768 NOTIFY
Contact: <sip:terminal.vancouver.example.com>
Content-Length: 0
```

5. Afin de servir l'abonnement, le RLS local s'abonne à l'état des ressources. Dans cette étape, le RLS tente de s'abonner à l'état de présence de la ressource "sip:ed@dallas.example.net". Comme le RLS local sait comment recevoir les notifications pour les abonnements de liste, il inclut le champ d'en-tête "Supported: eventlist" dans sa demande. Bien que le lien entre cet abonnement et celui envoyé par le terminal soit laissé à l'appréciation de l'application, ce message montre un comportement raisonnable en incluant des champs d'en-tête "Accept" pour tous les types de corps dont il sait qu'ils sont pris en charge par l'abonné (Terminal). Ceci est sûr à faire, car le RLS local va seulement passer ces formats à l'abonné et n'a pas en fait besoin de les comprendre.

Local RLS -> Serveur de présence à dallas.example.net

```
SUBSCRIBE sip:ed@dallas.example.net SIP/2.0
Via: SIP/2.0/TCP pres.vancouver.example.com;
  branch=z9hG4bKMEyGjdG1LH

Max-Forwards: 70
To: <sip:ed@dallas.example.net>
From: <sip:adam@vancouver.example.com>;tag=aM5icQu9
Call-ID: Ugwz5ARxNw@pres.vancouver.example.com
```

CSeq: 870936068 SUBSCRIBE
Contact: <sip:pres.vancouver.example.com>
Identity:
Tm8sIHRoaXMgaXNuJ3QgYSByZWFsIGNlcnQuIFlvdSBvnZpb3VzbHkgaGF2ZSB0aW1lIHRvIGtpbGwuIEkKc3VnZ2VzdCBodHRwOi8vd3d3LmhvbWVzdGFycnVubmVyLmNvbS8K
Identity-Info: https://vancouver.example.com/cert
Event: presence
Expires: 3600
Supported: eventlist
Accept: application/pidf+xml
Accept: application/rlmi+xml
Accept: multipart/related
Accept: multipart/signed
Accept: application/pkcs7-mime
Content-Length: 0

6. Le serveur de présence à dallas.example.net termine la transaction SUBSCRIBE. Noter que l'authentification devrait normalement prendre place à ce point du flux d'appels. Cette étape est omise pour faire bref.

Serveur de présence à dallas.example.net -> Local RLS

SIP/2.0 200 OK
Via: SIP/2.0/TCP pres.vancouver.example.com;
 branch=z9hG4bKMEyGjdG1LH
To: <sip:ed@dallas.example.net>;tag=e45TmHTh
From: <sip:adam@vancouver.example.com>;tag=aM5icQu9
Call-ID: Ugwz5ARxNw@pres.vancouver.example.com
CSeq: 870936068 SUBSCRIBE
Contact: <sip:dallas.example.net>
Expires: 3600
Content-Length: 0

7. Dans cet exemple, on suppose que le serveur à dallas.example.net n'a pas assez d'informations d'autorisation pour rejeter ou accepter notre abonnement. Le notify initial, contient donc un "Subscription-State" de "pending". On peut supposer que la partie responsable de l'acceptation ou du refus d'autorisation pour la ressource est notifiée de ce changement ; cependant, ces étapes ne sont pas incluses dans ce flux d'appel pour faire bref.

Serveur de présence à dallas.example.net -> RLS local

NOTIFY sip:pres.vancouver.example.com SIP/2.0
Via: SIP/2.0/TCP pres.dallas.example.net;
 branch=z9hG4bKfwpklPxmrW
Max-Forwards: 70
From: <sip:ed@dallas.example.net>;tag=e45TmHTh
To: <sip:adam@vancouver.example.com>;tag=aM5icQu9
Call-ID: Ugwz5ARxNw@pres.vancouver.example.com
CSeq: 1002640632 NOTIFY
Contact: <sip:dallas.example.net>
Subscription-State: pending;expires=3600
Event: presence
Require: eventlist
Content-Length: 0

8. Le RLS local termine la transaction NOTIFY. Noter que, à ce point, le RLS local a de nouvelles informations à rapporter à l'abonné. Le choix de rapporter immédiatement les informations ou de les garder pour les livrer ultérieurement appartient complètement à l'application. Pour cet exemple, on suppose que le RLS va attendre un court instant avant de le faire, afin de permettre que les abonnements qu'il a envoyé aient suffisamment de temps pour fournir des données utiles.

RLS local -> Serveur de présence à dallas.example.net

SIP/2.0 200 OK

```
Via: SIP/2.0/TCP pres.dallas.example.net;
  branch=z9hG4bKfwpklPxmrW
From: <sip:ed@dallas.example.net>;tag=e45TmHTh
To: <sip:adam@vancouver.example.com>;tag=aM5icQu9
Call-ID: Ugwz5ARxNw@pres.vancouver.example.com
CSeq: 1002640632 NOTIFY
Contact: <sip:pres.vancouver.example.com>
Content-Length: 0
```

9. Le RLS local s'abonne à l'état de l'autre ressource non locale.

RLS local -> RLS à stockholm.example.org

```
SUBSCRIBE sip:adam-friends@stockholm.example.org SIP/2.0
Via: SIP/2.0/TCP pres.vancouver.example.com;
  branch=z9hG4bKFSrAF8CZFL
Max-Forwards: 70
To: <sip:adam-friends@stockholm.example.org>
From: <sip:adam@vancouver.example.com>;tag=a12eztNf

Call-ID: kBq5XhtZLN@pres.vancouver.example.com
CSeq: 980774491 SUBSCRIBE
Contact: <sip:pres.vancouver.example.com>
Identity: Tm90IGEgcmVhbCBzaWduYXR1cmUsIGVpdGhlci4gQ2VydGFp
      bmx5IHlvdSBoYXZlIGJldHRlcgp0aGluZ3MgdG8gYmUgZG9p
      bmcuIEhhdmUgeW91IGZpbmlzaGVkIHlvdXIgUkxTIHlldD8K
Identity-Info: https://vancouver.example.com/cert
Event: presence
Expires: 3600
Supported: eventlist
Accept: application/pidf+xml
Accept: application/rlmi+xml
Accept: multipart/related
Accept: multipart/signed
Accept: application/pkcs7-mime
Content-Length: 0
```

10. Le RLS à stockholm.example.org termine la transaction SUBSCRIBE. Noter que l'authentification prendrait normalement place à ce point du flux d'appel. Cette étape est omise pour faire bref.

RLS à stockholm.example.org -> RLS local

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP pres.vancouver.example.com;
  branch=z9hG4bKFSrAF8CZFL
To: <sip:adam-friends@stockholm.example.org>;tag=JenZ40P3
From: <sip:adam@vancouver.example.com>;tag=a12eztNf
Call-ID: kBq5XhtZLN@pres.vancouver.example.com
CSeq: 980774491 SUBSCRIBE
Contact: <sip:stockholm.example.org>
Expires: 3600
Content-Length: 0
```

11. Dans cet exemple, on suppose que le RLS à stockholm.example.org est aussi une autorité pour les informations de présence pour les utilisateurs dans le domaine "stockholm.example.org". Le NOTIFY contient un document RLMI qui décrit la liste des amis contenue, ainsi que les informations de présence pour ces utilisateurs. Dans ce cas particulier, le RLS à stockholm.example.org a choisi de signer [RFC1847] le corps du message NOTIFY. Comme décrit dans la RFC 3851, la signature est effectuée en créant un document multipart/signed qui a deux parties. La première partie est le document à signer (dans cet exemple, le document multipart/related qui décrit les états de la liste de ressources) tandis que la seconde partie est la signature réelle.

RLS à stockholm.example.org -> RLS local

```
NOTIFY sip:pres.vancouver.example.com SIP/2.0
Via: SIP/2.0/TCP pres.stockholm.example.org;
  branch=z9hG4bKmGL1nyZfQI
Max-Forwards: 70
From: <sip:adam-friends@stockholm.example.org>;tag=JenZ40P3
To: <sip:adam@vancouver.example.com>;tag=a12eztNf
Call-ID: kBq5XhtZLN@pres.vancouver.example.com
CSeq: 294444656 NOTIFY
Contact: <sip:stockholm.example.org>
Event: presence
Subscription-State: active;expires=3600
Require: eventlist
Content-Type: multipart/signed;
   protocol="application/pkcs7-signature";
   micalg=sha1;boundary="l3WMZaaL8NpQWGnQ4mlU"
Content-Length: 2038

--l3WMZaaL8NpQWGnQ4mlU
Content-Transfer-Encoding: binary
Content-ID: <ZPvJHL@stockholm.example.org>
Content-Type: multipart/related;type="application/rlmi+xml";
   start="<Cvjpeo@stockholm.example.org>";
   boundary="tuLLl3lDyPZX0GMr2YOo"

--tuLLl3lDyPZX0GMr2YOo
Content-Transfer-Encoding: binary
Content-ID: <Cvjpeo@stockholm.example.org>
Content-Type: application/rlmi+xml;charset="UTF-8"

<?xml version="1.0" encoding="UTF-8"?>
<list xmlns="urn:ietf:params:xml:ns:rlmi"
     uri="sip:adam-friends@stockholm.example.org" version="1"
     fullState="true">
  <name xml:lang="en">Buddy List at COM</name>
  <name xml:lang="de">Liste der Freunde an COM</name>
  <ressource uri="sip:joe@stockholm.example.org">
    <name>Joe Thomas</name>
    <instance id="1" state="active"
          cid="mrEakg@stockholm.example.org"/>
  </ressource>
  <ressource uri="sip:mark@stockholm.example.org">
    <name>Mark Edwards</name>
    <instance id="1" state="active"
          cid="KKMDmv@stockholm.example.org"/>
  </ressource>
</list>

--tuLLl3lDyPZX0GMr2YOo
Content-Transfer-Encoding: binary
Content-ID: <mrEakg@stockholm.example.org>
Content-Type: application/pidf+xml;charset="UTF-8"

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
   entity="sip:joe@stockholm.example.org">
  <tuple id="x823a4">
    <status>
      <basic>open</basic>
    </status>
```

```
    <contact priority="1.0">sip:joe@stockholm.example.org</contact>
  </tuple>
</presence>

--tuLLl3lDyPZX0GMr2YOo
Content-Transfer-Encoding: binary
Content-ID: <KKMDmv@stockholm.example.org>
Content-Type: application/pidf+xml;charset="UTF-8"

<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    entity="sip:mark@stockholm.example.org">
  <tuple id="z98075">
    <status>
      <basic>closed</basic>
    </status>
  </tuple>
</presence>

--tuLLl3lDyPZX0GMr2YOo--

--l3WMZaaL8NpQWGnQ4mlU
Content-Transfer-Encoding: binary
Content-ID: <K9LB7k@stockholm.example.org>
Content-Type: application/pkcs7-signature

[la signature PKCS n° 7 vient ici]

--l3WMZaaL8NpQWGnQ4mlU--
```

12. Le RLS local termine la transaction NOTIFY.

RLS local -> RLS à stockholm.example.org

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP pres.stockholm.example.org;
  branch=z9hG4bKmGL1nyZfQI
From: <sip:adam-friends@stockholm.example.org>;tag=JenZ40P3
To: <sip:adam@vancouver.example.com>;tag=a12eztNf
Call-ID: kBq5XhtZLN@pres.vancouver.example.com
CSeq: 294444656 NOTIFY
Contact: <sip:pres.vancouver.example.com>
Content-Length: 0
```

13. À ce point, le RLS local décide qu'il a collecté assez d'informations supplémentaires pour effectuer une nouvelle notification à l'utilisateur. Bien que l'envoi d'une pleine notification serait parfaitement acceptable, le RLS décide d'envoyer plutôt une notification partielle. Le document RLMI contient seulement les informations sur les ressources mises à jour, comme indiqué par le réglage du paramètre "fullState" à "faux". Pour éviter de corrompre la signature S/MIME sur les données reçues du RLS à stockholm.example.org, le RLS local copie le corps multipart/signed entier tel quel dans la notification qu'il envoie.

RLS Local -> Terminal

```
NOTIFY sip:terminal.vancouver.example.com SIP/2.0
Via: SIP/2.0/TCP pres.vancouver.example.com;
  branch=z9hG4bK4EPlfSFQK1
Max-Forwards: 70
From: <sip:adam-buddies@pres.vancouver.example.com>;tag=zpNctbZq
To: <sip:adam@vancouver.example.com>;tag=ie4hbb8t
Call-ID: cdB34qLToC@terminal.vancouver.example.com
CSeq: 997935769 NOTIFY
```

```
Contact: <sip:pres.vancouver.example.com>
Event: presence
Subscription-State: active;expires=7200
Require: eventlist
Content-Type: multipart/related;type="application/rlmi+xml";
   start="<2BEI83@pres.vancouver.example.com>";
   boundary="TfZxoxgAvLqgj4wRWPDL"
Content-Length: 2862

--TfZxoxgAvLqgj4wRWPDL
Content-Transfer-Encoding: binary
Content-ID: <2BEI83@pres.vancouver.example.com>
Content-Type: application/rlmi+xml;charset="UTF-8"

<?xml version="1.0" encoding="UTF-8"?>
<list xmlns="urn:ietf:params:xml:ns:rlmi"
      uri="sip:adam-friends@pres.vancouver.example.com" version="2"
      fullState="false">
  <name xml:lang="en">Buddy List at COM</name>
  <name xml:lang="de">Liste der Freunde an COM</name>
  <ressource uri="sip:ed@dallas.example.net">
    <name>Ed at NET</name>
    <instance id="sdlkmeopdf" state="pending"/>
  </ressource>
  <ressource uri="sip:adam-friends@stockholm.example.org">
    <name xml:lang="en">My Friends at ORG</name>
    <name xml:lang="de">Meine Freunde an ORG</name>
    <instance id="cmpqweitlp" state="active"
            cid="1KQhyE@pres.vancouver.example.com"/>
  </ressource>
</list>

--TfZxoxgAvLqgj4wRWPDL
Content-Transfer-Encoding: binary
Content-ID: <1KQhyE@pres.vancouver.example.com>
Content-Type: multipart/signed;
   protocol="application/pkcs7-signature";
   micalg=sha1;boundary="l3WMZaaL8NpQWGnQ4mlU"

--l3WMZaaL8NpQWGnQ4mlU
Content-Transfer-Encoding: binary
Content-ID: <ZPvJHL@stockholm.example.org>
Content-Type: multipart/related;type="application/rlmi+xml";
   start="<Cvjpeo@stockholm.example.org>";
   boundary="tuLLl3lDyPZX0GMr2YOo"

--tuLLl3lDyPZX0GMr2YOo
Content-Transfer-Encoding: binary
Content-ID: <Cvjpeo@stockholm.example.org>
Content-Type: application/rlmi+xml;charset="UTF-8"
<?xml version="1.0" encoding="UTF-8"?>
<list xmlns="urn:ietf:params:xml:ns:rlmi"
      uri="sip:adam-friends@stockholm.example.org" version="1"
      fullState="true">
  <name xml:lang="en">Buddy List at ORG</name>
  <name xml:lang="de">Liste der Freunde an ORG</name>
  <ressource uri="sip:joe@stockholm.example.org">
    <name>Joe Thomas</name>
    <instance id="1" state="active"
            cid="mrEakg@stockholm.example.org"/>
  </ressource>
```

```
  <ressource uri="sip:mark@stockholm.example.org">
    <name>Mark Edwards</name>
    <instance id="1" state="active"
          cid="KKMDmv@stockholm.example.org"/>
  </ressource>
</list>

--tuLLl3lDyPZX0GMr2YOo
Content-Transfer-Encoding: binary
Content-ID: <mrEakg@stockholm.example.org>
Content-Type: application/pidf+xml;charset="UTF-8"
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    entity="sip:joe@stockholm.example.org">
  <tuple id="x823a4">
    <status>
      <basic>open</basic>
    </status>
    <contact priority="1.0">sip:joe@stockholm.example.org</contact>
  </tuple>
</presence>

--tuLLl3lDyPZX0GMr2YOo
Content-Transfer-Encoding: binary
Content-ID: <KKMDmv@stockholm.example.org>
Content-Type: application/pidf+xml;charset="UTF-8"
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    entity="sip:mark@stockholm.example.org">
  <tuple id="z98075">
    <status>
      <basic>closed</basic>
    </status>
  </tuple>
</presence>
--tuLLl3lDyPZX0GMr2YOo--

--l3WMZaaL8NpQWGnQ4mlU
Content-Transfer-Encoding: binary
Content-ID: <K9LB7k@stockholm.example.org>
Content-Type: application/pkcs7-signature

[La signature PKCS n° 7 vient ici]

--l3WMZaaL8NpQWGnQ4mlU--

--TfZxoxgAvLqgj4wRWPDL--
```

14. Le terminal termine la transaction NOTIFY.

Terminal -> RLS local

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP pres.vancouver.example.com;
  branch=z9hG4bK4EPlfSFQK1
From: <sip:adam-buddies@pres.vancouver.example.com>;tag=zpNctbZq
To: <sip:adam@vancouver.example.com>;tag=ie4hbb8t
Call-ID: cdB34qLToC@terminal.vancouver.example.com
CSeq: 997935769 NOTIFY
Contact: <sip:terminal.vancouver.example.com>
Content-Length: 0
```

7. Considérations pour la sécurité

Noter que les mécanismes pour obtenir des informations d'état pour les ressources dans une liste sont généralement laissés au choix de la mise en œuvre de RLS. Certaines des questions de sécurité ci-dessous sont spécifiques des circonstances dans lesquelles un SIP abonnement spécialisé est utilisé à de telles fins. Des mécanismes non SIP pour obtenir des informations d'état sur les ressources d'une liste vont normalement avoir leurs propres questions de sécurité associées pour le faire ; cependant, énumérer exhaustivement de telles méthodes d'accès n'est pas possible dans ce document. Les mises en œuvre qui utilisent de tels mécanismes doivent analyser les méthodes d'accès choisies quant aux questions de sécurité pertinentes.

7.1 Authentification

Si des abonnements spécialisés sont nécessaires pour restituer des informations d'état de ressource, l'utilisateur final n'est plus l'abonné direct à l'état de la ressource. Cela signifie que l'authentification directe de l'utilisateur n'est plus possible.

7.1.1 RLS et abonné dans le même domaine

On s'attend à ce que le déploiement le plus courant des RLS entraîne que les abonnés au RLS vont être dans le même domaine que le RLS. Quand c'est le cas, le RLS a alors la capacité d'agir comme un service d'authentification. Le rôle d'un service d'authentification est défini dans la [RFC4474] "Améliorations de la gestion d'identité authentifiée dans le protocole d'initialisation de session (SIP)".

En général, dans ce système, le RLS authentifie l'abonné et ensuite inclut un champ d'en-tête "Identité" dans tous les abonnements spécialisés effectués au nom de cet utilisateur authentifié. Ce champ d'en-tête "Identité" affirme cryptographiquement que la demande a été autorisée au nom de l'utilisateur indiqué dans le champ d'en-tête "From".

Parce que la capacité d'authentifier les demandes est centrale pour le bon fonctionnement du réseau, tout RLS qui utilise les abonnements spécialisés SIP pour acquérir des informations sur les ressources d'une liste de ressources DOIT être capable d'agir comme un service d'authentification comme défini dans la [RFC4474], pourvu que la politique administrative locale lui permette de le faire.

En d'autres termes, toutes les mises en œuvre de RLS qui prennent en charge les abonnements SIP spécialisés doivent aussi inclure la capacité d'être configurées à agir comme service d'authentification. Il appartient entièrement à un administrateur de choisir d'activer ou non une telle caractéristique. Bien sûr, sans la capacité d'agir comme serveur d'identité, tout RLS ainsi configuré va se comporter comme décrit au paragraphe suivant, car il agit effectivement comme si il était dans un domaine différent de celui de l'utilisateur.

7.1.2 RLS et abonné dans des domaines différents

En général, les extensions SIP d'identité authentifiée ne donnent pas de moyen pour que le RLS affirme en toute sécurité que les abonnements sont effectués au nom de l'utilisateur final. Précisément, quand l'abonné et le RLS sont dans des domaines différents, le RLS n'aura aucun moyen pour affirmer l'identité de l'utilisateur. Les mécanismes par lesquels les abonnements spécialisés peuvent être authentifiés dans de telles circonstances feront l'objet d'études ultérieures.

Jusqu'à ce que de telles solutions générales soient développées, les RLS qui sont dans un domaine différent de celui de l'abonné au nom duquel ils créent des abonnements spécialisés DEVRAIENT s'abonner aux ressources en utilisant leur propre identité. Ce faisant, le RLS va généralement obtenir seulement les informations de ressource qui sont publiquement disponibles.

En l'absence de telles solutions générales, les mises en œuvre d'agents d'utilisateur d'abonné PEUVENT tenter des abonnements directs aux ressources de la liste de ressources quand ils s'abonnent à un RLS en dehors de leur domaine (soit directement soit au moyen d'un autre abonnement de liste de ressources). Les ressources objet de l'abonnement vont être celles indiquées dans l'attribut "uri" des éléments <ressource> présents dans le document RLMI retourné par le RLS. S'abonner directement aux ressources permet qu'ait lieu une authentification appropriée de l'utilisateur, qui va généralement les autoriser à recevoir des informations d'état plus complètes. Les mises en œuvre qui choisissent d'effectuer de tels abonnements directs DEVRAIENT utiliser les données restituées plutôt que toute information sur la ressource obtenue via l'abonnement de liste.

7.2 Risques d'agrégation inappropriée

Un serveur de liste de ressources sert normalement des informations à plusieurs abonnés à la fois. Dans de nombreux cas, les ressources peuvent être présentes dans plusieurs listes ; de plus, il est possible que les serveurs de liste de ressources aient deux utilisateurs qui s'abonnent à la même liste.

Dans ces cas, des mises en œuvre de RLS mal guidées peuvent tenter de minimiser la charge du réseau en gardant seulement un abonnement spécialisé à une ressource dans une liste et présenter le résultat d'un tel abonnement à plus d'un utilisateur. Bien sûr, faire ainsi circonvient toute politique d'autorisation qu'a le notificateur pour la ressource. On se souviendra que l'autorisation est souvent beaucoup plus qu'une simple décision binaire "permis/non permis" ; les ressources peuvent rendre des états de ressource très différents -- et même en conflit – selon l'identité de l'utilisateur qui s'abonne.

Pour empêcher la transmission d'informations d'événement à tout autre que le receveur prévu, les mises en œuvre NE DOIVENT PAS présenter le résultat d'un abonnement spécialisé à plus d'un utilisateur, à moins que :

a. le RLS ait un accès adéquat à la politique d'autorisation complète associée à la ressource à laquelle l'abonnement spécialisé a été fait, ET
b. le RLS puiss et ait déterminé que présenter les informations à plus d'un utilisateur ne viole pas une telle politique.

Noter que c'est un problème très difficile à résoudre correctement. Même dans les cas où un tel accès est estimé possible, ce mode de fonctionnement est NON RECOMMANDÉ.

7.3 Signature et sceau

Les mises en œuvre devraient se souvenit que toute section de corps MIME peut être signée et/ou chiffrée comme nécessaire. Les RLS devraient veiller à ne pas modifier de corps MIME reçu de tout abonnement spécialisé, et ne devraient pas généralement s'appuyer sur le fait d'être capable de les lire.

Afin de faciliter la sécurité, les serveurs de liste de ressources DEVRAIENT passer l'indication de la prise en charge des types de contenu "multipart/signed" et "application/pkcs7-mime" à tous les abonnements spécialisés SIP, si l'abonné les inclut dans le message initial SUBSCRIBE. Ne pas le faire peut en fait résulter en ce que des ressources refusent de divulguer l'état (si la politique du notificateur exige le chiffrement, mais si le RLS échoue à porter la prise en charge) ou en ce que des abonnés éliminent un état valide (si la politique de l'abonné exige une signature, mais que le RLS échoue à porter la prise en charge).

Noter que la mise en œuvre réelle du chiffrement et de la signature par le RLS n'est pas nécessairement capable de passer des corps signés et/ou chiffrés.

7.4 Boucles infinies

Un risque introduit par la capacité d'incorporer des listes de ressources est la possibilité de créer des listes qui finalement se contiennent elles-mêmes comme sous liste. La détection et le traitement de ce cas est trivial quand le RLS dessert tous les abonnements virtuels en interne. Quand les abonnements spécialisés sont créés pour desservir des abonnements virtuels, la détection de ces situations devient cependant un problème plus difficile.

Les mises en œuvre de RLS qui créent des abonnements spécialisés DOIVENT utiliser des sauvegardes pour empêcher de telles incorporations de créer des boucles d'abonnements infinies. Normalement, de tels mécanismes vont exiger leur prise en charge dans le protocole d'abonnement spécialisé. En particulier, l'application de filtres aux abonnements spécialisés peut être un moyen efficace d'empêcher de tels problèmes.

8. Considérations relatives à l'IANA

8.1 Nouvelle étiquette d'option SIP : eventlist

Ce paragraphe définit une nouvelle étiquette d'option pour le registre établi par le paragraphe 27.1 de la [RFC3261].
Nom d'étiquette d'option : eventlist
Description : extension pour permettre les abonnements aux listes de ressources.
Spécification publiée : RFC 4662

8.2 Nouveau type MIME pour méta informations de liste de ressources

Nom de type de support MIME : application

Nom de sous type MIME : rlmi+xml
Paramètres exigés : aucun
Paramètres facultatifs : charset. Voir dans la [RFC3023] une discussion du paramètre charset sur les types MIME déduits de XML. Comme ce type MIME est utilisé exclusivement dans SIP, l'utilisation du codage UTF-8 est fortement encouragée.
Considérations de codage : texte 8 bits
Considérations de sécurité : les considérations de sécurité spécifiques de l'usage de ce type MIME sont discutées dans la RFC 4662. Les [RFC1874] et [RFC3023] discutent les questions de sécurité communes à tous les usages de XML.
Considérations d'interopérabilité : l'utilisation de ce corps MIME est destinée à être généralement interopérable. Aucune considération d'unicité n'a été identifiée.
Spécification publiée : RFC 4662
Applications qui utilisent ce type de support : ce type de support est utilisé pour porter des méta informations sur l'état des listes de ressources au sein d'un abonnement du protocole d'initialisation de session (SIP).
Information supplémentaires :
Numéro magique : aucun.
Extensions de fichier : aucune.
Code de type de fichier Macintosh : aucun.
Identifiant d'objet ou OID : aucun.
Usage prévu : limité
Autres informations/commentaires généraux : aucun.
Personne à contacter pour plus d'informations : Adam Roach, adam@estacado.net
Auteur/Contrôleur des changements : la spécification de ce type MIME est un produit du groupe de travail SIMPLE et ses auteurs sont Adam Roach, Jonathan Rosenberg, et Ben Campbell. L'IETF a le contrôle des changements de cette spécification.

8.3 URN de sous espace de noms

URI : urn:ietf:params:xml:ns:rlmi

Description : ceci est l'URi d'espace de noms XML pour les éléments XML définis par la RFC 4662 pour décrire les informations sur les abonnements quand de tels abonnements sont agrégés au sein d'un seul abonnement SIP. Il est utilisé dans le type de corps application/rlmi+xml.

Contact d'enregistrement : Adam Roach, adam@estacado.net
Auteur/Contrôleur des changements : la spécification de ce type MIME est un produit du groupe de travail SIMPLE et ses auteurs sont Adam Roach, Jonathan Rosenberg, et Ben Campbell. L'IETF a le contrôle des changements de cette spécification.

XML :
DÉBUT

```
<?xml version="1.0"?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
 <html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="content-type"
     content="text/html;charset=utf-8"/>
  <title>Namespace for SIP Event Resource List
        Meta-Information</title>
 </head>
 <body>
  <h1>Namespace for SIP Event Resource List
      Meta-Information</h1>
  <h2>application/rlmi+xml</h2>
  <p>See <a href="[http://www.rfc-editor.org/rfc/rfc4662.txt]">
     RFC4662</a>.</p>
 </body>
 </html>
```

FIN

9. Remerciements

Merci à Sean Olson de sa relecture et ses corrections à l'utilisation de XML dans ce protocole. Merci aussi à Hisham Khartabil, Paul Kyzivat, Keith Drage, et Robert Sparks de leur relecture attentive et de leurs commentaires sur ce document.

10. Références

10.1 Références normatives

[RFC2045] N. Freed et N. Borenstein, "Extensions de messagerie Internet multi-objets (MIME) Partie 1 : Format des corps de message Internet", novembre 1996. *(D. S., MàJ par* 2184, 2231, 5335.)

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

[RFC2387] E. Levinson, "Type de contenu MIME Multiparti/Relatif", août 1998. *(P.S.)*

[RFC3066] H. Alvestrand, "Étiquettes pour l'identification des langues", BCP 47, janvier 2001. *(Obsolète, voir la* RFC4646.)

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

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

[RFC4474] J. Peterson et C. Jennings, "Améliorations de la gestion d'identité authentifiée dans le protocole d'initialisation de session (SIP)", août 2006. *(P.S. ; Remplacée par* RFC8224)

10.2 Références pour information

[RFC1847] J. Galvin, S. Murphy, S. Crocker, N. Freed, "Sécurité multiparties pour MIME : multipartie/signée et multipartie/chiffrée", octobre 1995. *(P.S.)*

[RFC1874] E. Levinson, "Types de supports SGML", décembre 1995. *(Expérimentale)*

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

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

[RFC3851] B. Ramsdell, "Spécification du message d'extensions de messagerie Internet multi-objets/sécurisé (S/MIME) version 3.1", juillet 2004. *(Obsolète, voir* RFC5751)

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

[RFC3859] J. Peterson, "Profil commun pour les services de présence (CPP)", août 2004. *(P.S.)*

[RFC4483] E. Burger, éd., "Mécanismes pour le contenu indirect dans les messages du protocole d'initialisation de session (SIP)", mai 2006. *(P.S.)*

Adresse des auteurs

Adam Roach
Estacado Systems
USA
mél : adam@estacado.net

Ben Campbell
Estacado Systems
USA
mél : ben@estacado.net

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054-2711
USA
mél : jdrosen@cisco.com

Déclaration complète de droits de reproduction

Propriété intellectuelle
L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourraient être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur le répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr .

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf- ipr@ietf.org .

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