Groupe de travail Réseau
Request for Comments : 3921
Catégorie : En cours de normalisation

P. Saint-Andre,éd., Jabber Software Foundation
octobre 2004
Traduction Claude Brière de L'Isle

Protocole extensible de messagerie et de présence (XMPP) : messagerie instantanée et présence

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.

Résumé

Le présent mémoire décrit des extensions et des applications des caractéristiques centrales du protocole extensible de messagerie instantanée et de présence (XMPP, *Extensible Messaging and Presence Protocol*) qui assurent les fonctions de base de la messagerie instantanée (IM) et de présence définies dans la RFC 2779.

Table des Matières

1. Introduction

1.1 Vue d'ensemble

Le protocole extensible de messagerie instantanée et de présence (XMPP, *Extensible Messaging and Presence Protocol*) est un protocole pour écouler des éléments XML [XML] afin d'échanger des informations de messagerie instantanée et de présence presque en temps réel. Les caractéristiques centrales de XMPP sont définies dans la [RFC3920] "Protocole extensible de messagerie instantanée et de présence (XMPP) : caractéristiques centrales". Ces caractéristiques, principalement des flux XML, utilisent TLS et SASL, et les enfants <message/>, <presence/>, et <iq/> de la racine de flux, fournissent les briques de construction de nombreux types d'applications presque en temps réel, qui peuvent être mises en couche par dessus ce cœur en envoyant des données spécifiques de l'application qualifiées par des espaces de noms XML particuliers [XML-NAMES]. Le présent mémoire décrit les extensions aux caractéristiques et aux applications du cœur de XMPP qui assurent les fonctionnalités de base attendues d'une application de messagerie instantanée (IM, *instant messaging*) et de présence comme défini par la [RFC2779].

1.2 Exigences

Pour les besoins du présent mémoire, les exigences de base d'une application de messagerie instantanée et de présence sont définies par la [RFC2779], qui stipule dans les grandes lignes les cas d'utilisation qu'un usager doit être capable de réaliser :

- Échanger des messages avec les autres usagers.
- Échanger des informations de présence avec les autres usagers.
- Gérer les abonnements des et aux autres usagers.
- Gérer les éléments d'une liste de contacts (qu'on appelle un "rôle" dans XMPP).
- Bloquer les communications de ou vers d'autres usagers spécifiques.

Les définitions détaillées de ces zones de fonctionnalités sont contenues dans la [RFC2779], et le lecteur intéressé est invité à consulter ce document pour ce qui concerne les exigences visées ici.

La [RFC2779] stipule aussi que les services de présence doivent être séparables des services de messagerie instantanée ; c'est-à-dire qu'il doit être possible d'utiliser le protocole pour fournir un service de présence, un service de messagerie instantanée, ou les deux. Bien que le texte du présent mémoire suppose que les mises en œuvre et déploiements voudront offrir un service unifié de messagerie instantanée et de présence, il n'y a aucune exigence qu'un service doive offrir les deux services de présence et de messagerie instantanée, et le protocole rend possible l'offre séparée et distincte des services pour la présence et pour la messagerie instantanée.

Note : Bien que la messagerie instantanée et la présence fondée sur XMPP satisfasse aux exigences de la [RFC2779], elle n'était pas conçue explicitement pour cette spécification, car le protocole de base a évolué dans un processus de développement ouvert au sein de la communauté libre Jabber avant la rédaction de la RFC 2779. Noter aussi que bien que des protocoles visant de nombreux autres domaines de fonctionnalités aient été définis dans la communauté Jabber, ces protocoles ne sont pas inclus dans le présent mémoire parce qu'ils ne sont pas exigés par la [RFC2779].

1.3 Terminologie

Le présent mémoire hérite de la terminologie définie dans la [RFC3920].

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

2. Syntaxe des Strophes XML

La sémantique de base et les attributs communs des strophes XML qualifiées par les espaces de noms 'jabber:client' et 'jabber:server' sont définis dans la [RFC3920]. Cependant, ces espaces de noms définissent aussi divers éléments fils, ainsi que des valeurs pour l'attribut 'type' commun, qui sont spécifiques des applications de messagerie instantanée et de présence. Donc, avant de traiter des "cas d'utilisation" particuliers pour de telles applications, on décrit un peu plus avant la syntaxe des strophes XML, complétant par là l'exposé de la [RFC3920].

2.1 Syntaxe de message

Les strophes de message qualifiées par l'espace de noms 'jabber:client' ou 'jabber:server' sont utilisées pour "pousser" les informations jusqu'à une autre entité. Les usages communs dans les applications de messagerie instantanée incluent des messages seuls, des messages envoyés dans le contexte d'une conversation, des messages envoyés dans le contexte d'une salle de débat multi-utilisateurs, des gros titres et autres alertes, et des erreurs.

2.1.1 Types de message

L'attribut 'type' d'une strophe de message est RECOMMANDÉ ; si il est inclus, il spécifie le contexte conversationnel du message, fournissant donc une indication concernant sa présentation (par exemple, dans une interface graphique d'usager). Si il est inclus, l'attribut 'type' DOIT avoir une des valeurs suivantes :

chat Le message est envoyé dans le contexte d'une conversation entre deux personnes. Un client conforme DEVRAIT présenter le message dans une interface permettant une causerie entre deux parties, incluant un historique de conversation approprié.

error Une erreur s'est produite dans le message précédent envoyé par l'envoyeur (pour les détails concernant la syntaxe d'erreur de strophe, se reporter à la [RFC3920]). Un client conforme DEVRAIT présenter une interface appropriée qui informe l'envoyeur de la nature de l'erreur.

groupchat Le message est envoyé dans le contexte d'un environnement de causerie multi utilisateurs (similaire à celui de la [RFC1459]). Un client conforme DEVRAIT présenter le message dans une interface permettant une causerie de plusieurs à plusieurs entre les parties, incluant un rôle des parties dans la salle de débat et un historique de conversation approprié. La définition complète des protocoles de causerie de groupe fondée sur XMPP sort du domaine d'application du présent mémoire.

headline Le message est probablement généré par un service automatisé qui livre ou diffuse les contenus (nouvelles, sports, informations sur les marchés, etc.). Aucune réponse n'est attendue pour le message, et un client conforme DEVRAIT présenter le message dans une interface qui différencie de façon appropriée le message des messages autonomes, des sessions de causerie, ou des sessions de causerie de groupe (par exemple, en ne fournissant pas au receveur la possibilité de répondre).

normal Le message est un seul message qui est envoyé en dehors du contexte d'une conversation bilatérale ou d'une causerie de groupe, et auquel on s'attend à ce que le receveur réponde. Un client conforme DEVRAIT présenter le message dans une interface qui permet au receveur de répondre, mais sans historique de conversation.

Une application d'IM DEVRAIT prendre en charge tous les types de message ci-dessus ; si une application reçoit un message sans attribut 'type' ou si l'application ne comprend pas la valeur de l'attribut 'type' fourni, elle DOIT considérer que le message est du type "normal" (c'est-à-dire, "normal" est la valeur par défaut). Le type "error" DOIT être généré seulement en réponse à une erreur relative à un message reçu d'une autre entité.

Bien que l'attribut 'type' soit FACULTATIF, on considère qu'il est poli de refléter le type dans toute réponse à un message ; de plus, certaines applications spécialisées (par exemple, un service de causerie multi-utilisateurs) PEUVENT à leur discrétion mettre en application l'utilisation d'un type de message particulier (par exemple, type='groupchat').

2.1.2 Éléments fils

Comme décrit au paragraphe 2.4 "Espace de noms étendu", une strophe de message PEUT contenir tout élément fils d'un espace de noms approprié.

Conformément à la déclaration d'espace de noms par défaut, une strophe de message est qualifiée par défaut par l'espace de noms 'jabber:client' ou 'jabber:server', qui définit certains enfants admissibles des strophes de message. Si la strophe de message est du type "error", elle DOIT inclure une <error/> fille ; pour les détails, voir la [RFC3920]. Autrement, la strophe de message PEUT contenir un des éléments fils suivants sans déclaration explicite d'espace de noms :

1. <subject/> *(sujet)*
2. <body/> *(corps)*
3. <thread/> *(fil)*

2.1.2.1 Subject

L'élément <subject/> contient des données de caractères XML lisibles par l'homme qui spécifient le sujet du message.

L'élément <subject/> NE DOIT PAS posséder d'attribut, à l'exception de l'attribut 'xml:lang'. Plusieurs instances de l'élément <subject/> PEUVENT être incluses afin de fournir des versions de remplacement du même sujet, mais seulement si chaque instance possède un attribut 'xml:lang' avec une valeur de langage distincte. L'élément <subject/> NE DOIT PAS contenir de contenu mixte (comme défini au paragraphe 3.2.2 de [XML]).

2.1.2.2 Body

L'élément <body/> contient des données de caractères XML lisibles par l'homme qui spécifient le contenu textuel du message ; cet élément fils est normalement inclus mais est FACULTATIF. L'élément <body/> NE DOIT PAS posséder d'attribut, à l'exception de l'attribut 'xml:lang'. Plusieurs instances de l'élément <body/> PEUVENT être incluses mais seulement si chaque instance possède un attribut 'xml:lang' qui a une valeur de langage distincte. L'élément <body/> NE DOIT PAS contenir de contenu mixte (comme défini au paragraphe 3.2.2 de [XML]).

2.1.2.3 Thread

L'élément <thread/> contient des données de caractères XML non lisibles par l'homme qui spécifient un identifiant utilisé pour suivre le fil d'une conversation (auquel on se réfère parfois comme à une "session de messagerie instantanée") entre deux entités. La valeur de l'élément< thread/> est générée par l'envoyeur et DEVRAIT être recopiée dans toutes les réponses. Si il est utilisé, il DOIT être unique pour ce fil de conversation au sein du flux et DOIT être cohérent tout au long de cette conversation (un client qui reçoit un message provenant du même JID complet mais avec un identifiant de fil différent DOIT supposer que le message en question existe en dehors du contexte de conversation existant). L'utilisation de l'élément <thread/> est FACULTATIVE et n'est pas utilisée pour identifier les messages individuels, seulement les conversations. Une strophe de message NE DOIT PAS contenir plus d'un élément <thread/>. L'élément <thread/> NE DOIT PAS posséder d'attribut. La valeur de l'élément <thread/> DOIT être traitée comme opaque par les entités ; aucune signification sémantique ne peut en être déduite, et seules des comparaisons exactes peuvent en être faites. L'élément <thread/> NE DOIT PAS contenir de contenu mixte (comme défini au paragraphe 3.2.2 de [XML]).

2.2 Syntaxe de présence

Les strophes de présence sont utilisées qualifiées par l'espace de noms 'jabber:client' ou 'jabber:server' pour exprimer la disponibilité réseau actuelle d'une entité (hors ligne ou en ligne, avec les divers sous états de ce dernier et du texte descriptif facultatif défini par l'utilisateur) et pour notifier cette disponibilité aux autres entités. Les strophes de présence sont aussi utilisées pour négocier et gérer les abonnement à la présence des autres entités.

2.2.1 Types de présence

L'attribut 'type' d'une strophe de présence est FACULTATIF. Une strophe de présence qui ne possède pas d'attribut 'type' est utilisée pour signaler au serveur que l'envoyeur est en ligne et disponible pour la communication. Si il est inclus, l'attribut 'type' spécifie un manque de disponibilité, une demande de gestion d'abonnement à la présence d'une autre entité, une demande de présence actuelle d'une autre entité, ou une erreur relative à une strophe de présence envoyée précédemment. Si il est inclus, l'attribut 'type' DOIT avoir une des valeurs suivantes :
unavailable *(indisponible)* : signale que l'entité n'est plus disponible pour la communication.
subscribe *(s'abonner)* : l'envoyeur souhaite s'abonner à la présence du receveur.
subscribed *(abonné)* : l'envoyeur a autorisé le receveur à recevoir sa présence.
unsubscribe *(désabonner)* : l'envoyeur s'est désabonné de la présence de l'autre entité.
unsubscribed *(désabonné)* : la demande d'abonnement a été refusée ou un abonnement antérieurement accordé a été annulé.
probe *(sonde)* : demande de présence actuelle d'une entité ; DEVRAIT n'être générée que par un serveur au nom d'un usager.
error : une erreur s'est produite concernant le traitement ou la livraison d'une strophe de présence envoyée antérieurement.

La Section 5 "Échange d'informations de présence" et la Section 6 "Gestion des abonnements" contiennent des informations détaillées sur la sémantique de présence et le modèle d'abonnement utilisés dans le contexte des applications de messagerie instantanée et de présence fondées sur XMPP.

2.2.2 Éléments fils

Comme décrit au paragraphe 2.4 "Espaces de noms étendus", une strophe de présence PEUT contenir tout élément fils d'un espace de noms approprié.

Conformément à la déclaration d'espace de noms par défaut, une strophe de présence est qualifiée par défaut par l'espace de noms 'jabber:client' ou 'jabber:server', qui définit certains enfants admissibles de strophes de présence. Si la strophe de présence est de type "error", elle DOIT inclure une <error/> fille ; pour les détails, voir la [RFC3920]. Si la strophe de présence

ne possède pas d'attribut 'type', elle PEUT contenir un des éléments fils suivants (noter que le <status/> fils PEUT être envoyé dans une strophe de présence de type "unavailable" ou, pour des raisons historiques, "subscribe") :
1. <show/>
2. <status/>
3. <priority/>

2.2.2.1 Show

L'élément <show/> FACULTATIF contient des données de caractères XML non lisibles par l'homme qui spécifient l'état de disponibilité particulier d'une entité ou ressource spécifique. Une strophe de présence NE DOIT PAS contenir plus d'un élément <show/>. L'élément <show/> NE DOIT PAS posséder d'attribut. Si il est fourni, la valeur des données de caractères XML DOIT être une des suivantes (des types de disponibilité supplémentaires pourraient être définis par un élément fils d'un espace de noms approprié de la strophe de présence) :
away : l'entité ou ressource est temporairement absente.
chat : l'entité ou ressource est activement intéressée par les causeries.
dnd : l'entité ou ressource est occupée (dnd = "Do Not Disturb", *Ne pas déranger*).
xa : l'entité ou ressource est absente pour une longue période (xa = "eXtended Away").

Si aucun élément <show/> n'est produit, l'entité est supposée être en ligne et disponible.

2.2.2.2 Status

L'élément <status/> FACULTATIF contient des données de caractères XML qui spécifient une description en langage naturel de l'état de disponibilité. Il est normalement utilisé en conjonction avec l'élément <show/> pour donner une description détaillée d'un état de disponibilité (par exemple, "en réunion"). L'élément <status/> NE DOIT PAS posséder d'attribut , à l'exception de l'attribut 'xml:lang'. Plusieurs instances de l'élément <status/> PEUVENT être incluses mais seulement si chaque instance possède un attribut 'xml:lang' avec une valeur de langue distincte.

2.2.2.3 Priority

L'élément <priority/> FACULTATIF contient des données de caractères XML non lisibles par l'homme qui spécifient le niveau de priorité de la ressource. La valeur DOIT être un entier entre -128 et +127. Une strophe de présence NE DOIT PAS contenir plus d'un élément <priority/>. L'élément <priority/> NE DOIT PAS posséder d'attribut. Si aucune priorité n'est fournie, un serveur DEVRAIT considérer que la priorité est zéro. Pour les informations concernant la sémantique des valeurs de priorité dans l'acheminement des strophes au sein des applications de messagerie instantanée et de présence, se reporter à la Section 11 "Règles de traitement des strophes XML par le serveur".

2.3 Syntaxe de IQ

Les strophes IQ fournissent un mécanisme structuré de demande-réponse. La sémantique de base de ce mécanisme (par exemple, que l'attribut 'id' est EXIGÉ) est définie dans la [RFC3920], tandis que la sémantique spécifique requise pour réaliser des cas d'utilisation particuliers est définie dans tous les cas par un espace de noms étendu (paragraphe 2.4) (noter que les espaces de noms 'jabber:client' et 'jabber:server' ne définissent aucun enfant des strophes IQ autre que le <error/> commun). Le présent mémoire définit deux de ces espaces de noms étendus, un pour la gestion de rôle (Section 7) et l'autre pour le blocage de communication (Section 10) ; cependant, une strophe IQ PEUT contenir des informations structurées qualifiées par tout espace de noms étendu.

2.4 Espaces de nom étendu

Alors que les trois sortes de strophe XML définies dans l'espace de noms "jabber:client" ou "jabber:server" (avec leurs attributs et éléments fils) fournissent un niveau de base de fonctionnalités pour la messagerie instantanée et la présence, XMPP utilise des espaces de noms XML pour étendre les strophes afin de fournir des fonctions supplémentaires. Donc un message ou strophe de présence PEUT contenir un ou plusieurs éléments fils facultatifs qui spécifient un contenu qui étend la signification du message (par exemple, une version formatée en XHTML du corps de message) et une strophe IQ PEUT contenir un tel élément fils. Cet élément fils PEUT avoir un nom quelconque et DOIT posséder une déclaration d'espace de noms 'xmlns' (autre que "jabber:client", "jabber:server", ou "http://etherx.jabber.org/streams") qui définit toutes les données contenues dans l'élément fils.

La prise en charge de tout espace de noms étendu est FACULTATIVE de la part d'une mise en œuvre (à part les espaces de noms étendus définis ici). Si une entité ne comprend pas un tel espace de noms, le comportement attendu de l'entité dépend de si l'entité est (1) le receveur ou (2) une entité qui achemine la strophe au receveur :

Receveur : Si un receveur reçoit une strophe qui contient un élément fils qu'il ne comprend pas, il DEVRAIT ignorer ces données XML spécifiques, c'est-à-dire, il NE DEVRAIT PAS les traiter ou les présenter à un usager ou une application associée (si il en est). En particulier :

* Si une entité reçoit un message ou strophe de présence qui contient des données XML qualifiées par un espace de noms qu'elle ne comprend pas, la portion de la strophe qui est dans l'espace de noms inconnu DEVRAIT être ignorée.
* Si une entité reçoit une strophe de message dont seulement un élément fils est qualifié par un espace de noms qu'elle ne comprend pas, elle DOIT ignorer la strophe entière.
* Si une entité reçoit une strophe IQ de type "get" ou "set" contenant un élément fils qualifié par un espace de noms qu'elle ne comprend pas, l'entité DEVRAIT retourner une strophe IQ de type "error" avec une condition d'erreur de <service-unavailable/>.

Routeur : Si une entité d'acheminement (généralement un serveur) traite une strophe qui contient un élément fils qu'elle ne comprend pas, elle DEVRAIT ignorer les données XML associées en les passant intactes au receveur.

3. Établissement de session

La plupart des applications de messagerie instantanée et de présence fondées sur XMPP sont mises en œuvre via une architecture client-serveur qui exige d'un client qu'il établisse une session sur un serveur afin de s'engager dans les activités attendues de messagerie instantanée et présence. Cependant, il y a plusieurs préconditions qui DOIVENT être satisfaites avant qu'un client puisse établir une session de messagerie instantanée et présence. Ce sont :

1. L'authentification de flux : un client DOIT réaliser l'authentification de flux comme documenté dans la [RFC3920] avant de tenter d'établir une session ou d'envoyer des strophes XML.
2. Lien de ressource : après l'achèvement de l'authentification de flux, un client DOIT lier une ressource au flux afin que l'adresse du client soit de la forme <usager@domaine/ressource>, après quoi l'entité est maintenant dite être une "ressource connectée" dans la terminologie de la [RFC3920].

Si un serveur accepte les sessions, il DOIT inclure un élément <session/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-session' dans les caractéristiques de flux qu'il annonce à un client après l'achèvement de l'authentification de flux comme défini dans la [RFC3920]:

Le serveur annonce les caractéristiques d'établissement de session au client :

```
<stream:stream
   xmlns='jabber:client'
   xmlns:stream='http://etherx.jabber.org/streams'
   id='c2s_345'
   from='exemple.com'
   version='1.0'>
<stream:features>
  <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
  <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
</stream:features>
```

Étant ainsi informé que l'établissement de session est exigé (et après achèvement du lien de ressource) le client DOIT établir une session si il désire s'engager dans la fonction de messagerie instantanée et présence ; il réalise cette étape en envoyant au serveur une strophe IQ de type "set" contenant un élément fils <session/> vide qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-session' :

Étape 1 : Le client demande une session avec le serveur :

```
<iq to='exemple.com'
   type='set'
   id='sess_1'>
  <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
</iq>
```

Étape 2 : Le serveur informe le client que la session a été créée :

```
<iq from='exemple.com'
   type='result'
```

```
  id='sess_1'/>
```

À l'établissement d'une session, une ressource connectée (dans la terminologie de la [RFC3920]) est dite être une "ressource active".

Plusieurs conditions d'erreur sont possibles. Par exemple, le serveur peut rencontrer une condition interne qui l'empêche de créer la session, le nom d'utilisateur ou l'identité d'autorisation peut n'avoir pas les permissions pour créer une session, ou il peut y avoir déjà une ressource active associée à un identifiant de ressource du même nom.

Si le serveur rencontre une condition interne qui l'empêche de créer la session, il DOIT retourner une erreur.

Étape 2 (alt) : Le serveur répond par une erreur (erreur interne du serveur) :

```
<iq from='exemple.com' type='error' id='sess_1'>
 <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
 <error type='wait'>
  <internal-server-error
     xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
 </error>
</iq>
```

Si le nom d'utilisateur ou de ressource n'a pas la permission de créer une session, le serveur DOIT retourner une erreur (par exemple, "forbidden" *(interdit)*).

Étape 2 (alt) : Le serveur répond par une erreur (nom d'utilisateur ou ressource non autorisée à créer une session) :

```
<iq from='exemple.com' type='error' id='sess_1'>
 <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
 <error type='auth'>
  <forbidden
     xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
 </error>
</iq>
```

Si il y a déjà une ressource active du même nom, le serveur DOIT soit (1) terminer la ressource active et permettre la nouvelle session demandée, soit (2) interdire la nouvelle session demandée et maintenir la ressource active. Ce que retient le serveur dépend de la mise en œuvre, bien qu'il soit RECOMMANDÉ de mettre en œuvre le cas n° 1. Dans le cas n° 1, le serveur DEVRAIT envoyer une erreur de flux <conflict/> à la ressource active, terminer le flux XML et la connexion TCP sous-jacente pour la ressource active, et retourner une strophe IQ de type "result" (indiquant le succès) à la nouvelle session demandée. Dans le cas n° 2, le serveur DEVRAIT envoyer une erreur de strophe <conflict/> à la nouvelle session demandée mais conserver le flux XML pour cette connexion afin que la nouvelle session demandée ait l'opportunité de négocier un identifiant de ressource non conflictuel avant d'envoyer une autre demande d'établissement de session.

Étape 2 (alt) : Le serveur informe la ressource active existante du conflit de ressource (cas n° 1) :

```
<stream:error>
 <conflict xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
</stream:error>
</stream:stream>
```

Étape 2 (alt) : Le serveur informe la nouvelle session demandée du conflit de ressources (cas n° 2) :

```
<iq from='exemple.com' type='error' id='sess_1'>
 <session xmlns='urn:ietf:params:xml:ns:xmpp-session'/>
 <error type='cancel'>
  <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
 </error>
</iq>
```

Après l'établissement d'une session, un client DEVRAIT envoyer la présence initiale et demander son rôle comme décrit ci-dessous bien que ces actions soient FACULTATIVES.

Note : Avant de permettre la création de sessions de messagerie instantanée et présence, un serveur PEUT exiger le

provisionnement préalable d'un compte. Les méthodes possibles pour le provisionnement d'un compte incluent la création d'un compte par un administrateur de serveur ou l'enregistrement d'un compte dans la bande en utilisant l'espace de noms 'jabber:iq:register' ; cette dernière méthode sort du domaine d'application du présent mémoire, mais est documentée dans [JEP-0077], publié par la Fondation des logiciels Jabber [JSF].

4. Échange de messages

Échanger des messages est une utilisation de base de XMPP qui est provoquée lorsque un usager génère une strophe de message adressée à une autre entité. Comme défini à la Section 11 "Règles pour le traitement des strophes XML par le serveur", le serveur de l'envoyeur est responsable de la livraison du message au receveur désigné (si le receveur est sur le même serveur) ou de l'acheminement du message au serveur du receveur (si le receveur est sur un serveur différent).

Pour des informations sur la syntaxe des strophes de message ainsi que sur leurs attributs et éléments fils définis, se référer à la syntaxe de message (paragraphe 2.1).

4.1 Spécifier un receveur désigné

Un client de messagerie instantanée DEVRAIT spécifier un receveur désigné pour un message en fournissant le JID d'une entité autre que l'envoyeur dans l'attribut 'to' de la strophe <message/>. Si le message est envoyé en réponse à un message reçu précédemment d'une adresse de forme <usager@domaine/ressource> (par exemple, dans le contexte d'une session de débat) la valeur de l'adresse 'to' DEVRAIT être de la forme <usager@domaine/ressource> plutôt que de la forme <usager@domaine> sauf si l'envoyeur a connaissance (via présence) que la ressource du receveur désigné n'est plus disponible. Si le message est envoyé en dehors du contexte d'une session de débat existante ou d'un message reçu, la valeur de l'adresse 'to' DEVRAIT être de forme <usager@domaine> plutôt que de la forme <usager@domaine/ressource>.

4.2 Spécifier un type de message

Comme on l'a noté, il est RECOMMANDÉ qu'une strophe de message possède un attribut 'type' dont la valeur comporte le contexte conversationnel (s'il en est) du message (voir"Type" (paragraphe 2.1.1)).

L'exemple suivant montre une valeur valide de l'attribut 'type' :

Exemple : Message d'un type défini :

```
<message
    to='romeo@exemple.net'
    from='juliet@exemple.com/balcony'
    type='chat'
    xml:lang='fr'>
  <body>Comment est tu, Romeo ?</body>
</message>
```

4.3 Spécifier un corps de message

Une strophe de message PEUT (et souvent va) contenir un élément <body/> fils dont les données de caractères XML spécifient la signification principale du message (voir le paragraphe 2.1.2.2 "Corps").

Exemple : Message avec un corps :

```
<message
    to='romeo@exemple.net'
    from='juliet@exemple.com/balcony'
    type='chat'
    xml:lang='fr'>
  <body>Comment vas tu, Romeo ?</body>
  <body xml:lang='cz'>Pro&#x010D;e&#x017D; jsi ty, Romeo?</body>
</message>
```

4.4 Spécifier un sujet de message

Une strophe de message PEUT contenir un ou plusieurs éléments <subject/> fils spécifiant le sujet du message (voir le paragraphe 2.1.2.1 "Sujet").

Exemple : Message avec un sujet :

```
<message
    to='romeo@exemple.net'
    from='juliet@exemple.com/balcony'
    type='chat'
    xml:lang='fr'>
  <subject>Je t'implore!</subject>
  <subject
      xml:lang='cz'>&#x00DA;p&#x011B;nliv&#x011B; prosim!</subject>
  <body>Comment vas tu, Romeo ?</body>
  <body xml:lang='cz'>Pro&#x010D;e&#x017D; jsi ty, Romeo?</body>
</message>
```

4.5 Spécifier un fil de conversation

Une strophe de message PEUT contenir un élément <thread/> fils qui spécifie le fil conversationnel dans lequel le message est situé, pour les besoins du traçage de la conversation (voir le paragraphe 2.1.2.3 "Fil").

Exemple : Conversation avec un fil :

```
<message
    to='romeo@exemple.net/orchard'
    from='juliet@exemple.com/balcony'
    type='chat'
    xml:lang='fr'>
  <body>N'est tu pas Roméo, et un Montaigu ?</body>
  <thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
</message>

<message
    to='juliet@exemple.com/balcony'
    from='romeo@exemple.net/orchard'
    type='chat'
    xml:lang='en'>
  <body>Neither, fair saint, if either thee dislike.</body>
  <thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
</message>

<message
    to='romeo@exemple.net/orchard'
    from='juliet@exemple.com/balcony'
    type='chat'
    xml:lang='en'>
  <body>How cam'st thou hither, tell me, et wherefore?</body>
  <thread>e0ffe42b28561960c6b12b944a092794b9683a38</thread>
</message>
```

5. Échange des informations de présence

L'échange des informations de présence est rendu relativement direct dans XMPP par l'utilisation des strophes de présence. Cependant, on voit ici un contraste avec le traitement de message : alors qu'un client PEUT envoyer des informations de présence dirigées à une autre entité en incluant une adresse 'to', normalement les notifications de présence (c'est-à-dire, des strophes de présence sans 'type' ou avec un type "unavailable" et sans adresse 'to') sont envoyées d'un client à son serveur et ensuite diffusées par le serveur à toute entité qui s'est abonnée à la présence de l'entité envoyeuse (dans la terminologie de la [RFC2778], ces entités sont des abonnés). Ce modèle de diffusion ne s'applique pas aux strophes de présence en rapport avec

l'abonnement ou aux strophes de présence de type "error", mais aux seules notifications de présence définies ci-dessus. (Noter que alors que les informations de présence PEUVENT être fournies au nom d'un usager par un service automatisé, il est normalement fourni par le client de l'usager.)

Pour des informations sur la syntaxe des strophes de présence ainsi que leurs attributs et éléments fils définis, se reporter à la [RFC3920].

5.1 Responsabilités du client et du serveur pour Présence

5.1.1 Présence initiale

Après l'établissement d'une session, un client DEVRAIT envoyer une présence initiale au serveur afin de signaler sa disponibilité pour des communications. Comme défini ici, la strophe de présence initiale (1) NE DOIT PAS posséder d'adresse 'to' (signalant qu'elle est destinée à être diffusée par le serveur au nom du client) et (2) NE DOIT PAS posséder d'attribut 'type' (signalant la disponibilité de l'usager). Après l'envoi de la présence initiale, une ressource active est dite être une "ressource disponible".

À réception d'une présence initiale d'un client, le serveur de l'usager DOIT faire ce qui suit si il n'y a pas déjà une ou plusieurs ressources disponibles pour l'usager (si il y a déjà une ou plusieurs ressources disponibles pour l'usager, le serveur n'a évidemment pas besoin d'envoyer les sondes de présence, car il possède déjà les informations requises) :

1. Envoyer des sondes de présence (c'est-à-dire, des strophes de présence dont l'attribut 'type' est réglé à une valeur de "probe") à partir du JID complet (par exemple, <usager@exemple.com/ressource>) de l'usager à tous les contacts auxquels l'usager est abonné afin de déterminer si ils sont disponibles ; de tels contacts sont ceux pour lesquels un JID est présent dans le rôle de l'usager avec l'attribut 'subscription' réglé à une valeur de "to" ou "both". (Noter que le serveur de l'usager NE DOIT PAS envoyer de sondes de présence aux contacts pour lesquels l'usager bloque les notifications de présence entrantes, comme décrit au paragraphe 10.10 "Blocage des notifications de présence entrantes".)

2. Diffuser une présence initiale à partir du JID complet (par exemple, <usager@exemple.com/ressource>) de l'usager à tous les contacts qui sont abonnés aux informations de présence de l'usager ; ces contacts sont ceux pour lesquels un JID est présent dans le rôle de l'usager avec l'attribut 'subscription' réglé à une valeur de "from" ou "both". (Noter que le serveur de l'usager NE DOIT PAS diffuser la présence initiale aux contacts pour lesquels l'usager bloque les notifications de présence sortantes, comme décrit au paragraphe 10.11 "Blocage des notifications de présence sortantes".)

De plus, le serveur de l'usager DOIT diffuser la présence initiale à partir de la nouvelle ressource disponible de l'usager à toutes les ressources disponibles existantes de l'usager (si il y en a).

À réception de la présence initiale venant de l'usager, le serveur du contact DOIT livrer la strophe de présence de l'usager aux JID complets (<contact@exemple.org/ressource>) associés à toutes les ressources disponibles du contact, mais seulement si l'usager est dans le rôle du contact avec un état d'abonnement de "to" ou "both" et si le contact n'a pas bloqué les notifications de présence entrantes provenant du JID nu ou complet de l'usager (comme défini au paragraphe 10.10 "Blocage des notifications de présence entrantes).

Si le serveur de l'usager reçoit une strophe de présence de type "error" en réponse à la présence initiale qu'il a envoyée à un contact au nom de l'usager, il NE DEVRAIT PAS envoyer d'autre mise à jour de présence à ce contact (jusqu'à ce qu'il reçoive une strophe de présence du contact).

5.1.2 Diffusion de présence

Après l'envoi d'une présence initiale, l'usager PEUT mettre à jour ses informations de présence pour les diffuser à tout moment durant sa session par l'envoi d'une strophe de présence sans adresse 'to' et soit pas d'attribut 'type', soit un attribut 'type' d'une valeur de "unavailable". (Noter qu'un client d'usager NE DEVRAIT PAS envoyer de mise à jour de présence pour diffuser des informations qui changent indépendamment de la présence et disponibilité de l'usager.)

Si la strophe de présence n'a pas d'attribut 'type' (c'est-à-dire, exprime la disponibilité) le serveur de l'usager DOIT diffuser le XML complet de cette strophe de présence à tous les contacts (1) qui sont dans le rôle de l'usager avec un type d'abonnement de "from" ou "both", (2) de qui l'usager n'a pas bloqué les notifications de présence sortantes, et (3) de qui le serveur n'a pas reçu une erreur de présence durant la session de l'usager (ainsi qu'à toutes les autres ressources disponibles de l'usager).

Si la strophe de présence a un attribut 'type' réglé à une valeur de "unavailable", le serveur de l'usager DOIT diffuser le XML complet de cette strophe de présence à toutes les entités qui entrent dans la description ci-dessus, ainsi qu'à toutes les entités

auxquelles l'usager a envoyé une présence disponible dirigée durant la session de l'usager (si l'usager n'a pas encore envoyé de présence indisponible dirigée à cette entité).

5.1.3 Sondes de présence

À réception d'une sonde de présence de l'usager, le serveur du contact DEVRAIT répondre somme suit :

1. Si l'usager n'est pas dans le rôle du contact avec un état d'abonnement de "From", "From + Pending Out", ou "Both" (comme défini à la Section 9 "États d'abonnement") le serveur du contact DOIT retourner une strophe de présence de type "error" en réponse à la sonde de présence (cependant, si un serveur reçoit une sonde de présence d'un sous domaine du nom d'hôte du serveur ou d'un autre service de confiance, il PEUT fournir des informations de présence sur l'usager à cette entité). Précisément :
   * Si l'usager est dans le rôle du contact avec un état d'abonnement de "None", "None + Pending Out", ou "To" (ou n'est pas du tout dans le rôle du contact) le serveur du contact DOIT retourner une erreur de strophe <forbidden/> en réponse à la sonde de présence.
   * Si l'usager est dans le rôle du contact avec un état d'abonnement de "None + Pending In", "None + Pending Out/In", ou "To + Pending In", le serveur du contact DOIT retourner une erreur de strophe <not-authorized/> en réponse à la sonde de présence.

2. Autrement, si le contact bloque les notifications de présence au JID nu ou au JID complet de l'usager (en utilisant soit une liste par défaut soit une liste active comme défini au paragraphe 10.11 "Blocage des notifications de présence sortantes") le serveur NE DOIT PAS répondre à la sonde de présence.

3. Autrement, si le contact n'a pas de ressource disponible, le serveur DOIT soit (1) répondre à la sonde de présence en envoyant à l'usager le XML complet de la dernière strophe de présence de type "unavailable" reçue du contact par le serveur, soit (2) ne pas répondre du tout.

4. Autrement, si le contact a au moins une ressource disponible, le serveur DOIT répondre à la sonde de présence par l'envoi à l'usager du XML complet de la dernière strophe de présence avec l'attribut 'to' reçu par le serveur provenant de chaque ressource disponible du contact (là encore, sous réserve des listes de confidentialité en vigueur pour chaque session).

5.1.4 Présence dirigée

Un usager PEUT envoyer une présence dirigée à une autre entité (c'est-à-dire, une strophe de présence avec un attribut 'to' dont la valeur est le JID de l'autre entité et avec soit pas d'attribut 'type', soit un attribut 'type' dont la valeur est "unavailable"). Il y a trois cas possibles :

1. Si l'usager envoie une présence dirigée à un contact qui est dans le rôle de l'usager avec un type d'abonnement de "from" ou "both" après avoir envoyé une présence initiale et avant d'envoyer une diffusion de présence indisponible, le serveur de l'usager DOIT acheminer ou livrer le XML complet de cette strophe de présence (sous réserve des listes de confidentialité) mais NE DEVRAIT PAS modifier autrement l'état du contact en ce qui concerne la diffusion de présence (c'est-à-dire, il DEVRAIT inclure le JID du contact dans toutes les diffusions suivantes de présence initiées par l'usager).

2. Si l'usager envoie une présence dirigée à une entité qui n'est pas dans le rôle de l'usager avec un type d'abonnement de "from" ou "both" après avoir envoyé une présence initiale et avant d'envoyer une diffusion de présence indisponible, le serveur de l'usager DOIT acheminer ou livrer le XML complet de cette strophe de présence à l'entité mais NE DOIT PAS modifier l'état du contact en ce qui concerne la diffusion de présence disponible (c'est-à-dire, il NE DOIT PAS inclure le JID de l'entité dans les diffusions suivantes de présence disponible initiées par l'usager) ; cependant, si la ressource disponible à partir de laquelle l'usager envoie la présence dirigée devient indisponible, le serveur de l'usager DOIT diffuser cette présence indisponible à l'entité (si l'usager n'a pas encore envoyé de présence indisponible dirigée à cette entité).

3. Si l'usager envoie une présence dirigée sans envoyer d'abord une présence initiale ou après avoir envoyé une diffusion de présence indisponible (c'est-à-dire, si la ressource est active mais non disponible) le serveur de l'usager DOIT traiter les entités auxquelles l'usager envoie la présence dirigée de la même façon qu'il traite les entités figurant dans le cas n° 2 ci-dessus.

5.1.5 Présence indisponible

Avant de terminer sa session avec un serveur, un client DEVRAIT devenir indisponible en douceur en envoyant une strophe de présence finale qui ne possède pas d'attribut 'to' et qui possède un attribut 'type' dont la valeur est "unavailable" (facultativement, la strophe de présence finale PEUT contenir un ou plusieurs éléments <status/> spécifiant la raison pour laquelle l'usager n'est plus disponible). Cependant, le serveur de l'usager NE DOIT PAS dépendre de la réception de la

présence finale d'une ressource disponible, car la ressource peut devenir indisponible de façon inattendue ou peut être arrêtée par le serveur. Si une des ressources de l'usager devient indisponible pour une raison quelconque (soit en douceur, soit autrement) le serveur de l'usager DOIT diffuser une présence indisponible à tous les contacts (1) qui sont sur le rôle de l'usager avec un type d'abonnement de "from" ou "both", (2) à qui l'usager n'a pas bloqué la présence sortante, et (3) de qui le serveur n'a pas reçu une erreur de présence durant la session de l'usager ; le serveur de l'usager DOIT aussi envoyer cette strophe de présence indisponible à toutes les autres ressources disponibles de l'usager, ainsi qu'à toutes les entités auxquelles l'usager a envoyé une présence dirigée durant la session de l'usager pour cette ressource (si l'usager n'a pas encore envoyé de présence indisponible dirigée à cette entité). Toute strophe de présence sans attribut 'type' et sans attribut 'to' qui est envoyée après l'envoi d'une présence indisponible dirigée ou la diffusion d('une présence indisponible DOIT être diffusée par le serveur à tous les abonnés.

5.1.6 Abonnements à Présence

Une demande d'abonnement est une strophe de présence dont l'attribut 'type' a une valeur de "subscribe". Si la demande d'abonnement est à envoyer à un contact de messagerie instantanée, le JID fourni dans l'attribut 'to' DEVRAIT être de la forme <contact@exemple.org> plutôt que <contact@exemple.org/ressource>, car le résultat désiré est normalement que l'usager reçoive la présence de toutes les ressources du contact, et pas simplement la ressource particulière spécifiée dans l'attribut 'to'.

Un serveur d'usager NE DOIT PAS approuver automatiquement les demandes d'abonnements au nom de l'usager. Toutes les demandes d'abonnement DOIVENT être dirigées sur le client de l'usager, spécifiquement sur une ou plusieurs ressources disponibles associées à l'usager. Si il n'y a pas de ressource disponible associée à l'usager lorsque la demande d'abonnement est reçue par le serveur de l'usager, le serveur de l'usager DOIT garder un enregistrement de la demande d'abonnement et livrer la demande lorsque l'usager crée ensuite une ressource disponible, jusqu'à ce que l'usager approuve ou refuse la demande. Si il y a plus d'une ressource disponible associée à l'usager lorsque la demande d'abonnement est reçue par le serveur de l'usager, le serveur de l'usager DOIT diffuser cette demande d'abonnement à toutes les ressources disponibles conformément aux "Règles de traitement des strophes XML par le serveur" (Section 11). (Noter que si une ressource active n'a pas fourni de présence initiale, le serveur NE DOIT PAS la considérer comme disponible et donc NE DOIT PAS lui envoyer de demande d'abonnement.) Cependant, si l'usager reçoit une strophe de présence de type "subscribe" d'un contact auquel l'usager a déjà accordé la permission de voir les informations de présence de l'usager (par exemple, dans des cas où le contact cherche à resynchroniser les états d'abonnement) le serveur de l'usager DEVRAIT auto répondre au nom de l'usager. De plus, le serveur de l'usager PEUT choisir de renvoyer au contact une demande d'abonnement en cours non approuvée, sur la base d'un algorithme spécifique de la mise en œuvre (par exemple, chaque fois qu'une nouvelle ressource devient disponible pour l'usager, ou après l'écoulement d'un certain délai) ; cela aide à récupérer d'erreurs transitoires silencieuses qui peuvent survenir en relation avec la demande d'abonnement originale.

5.2 Spécification de l'état de disponibilité

Un client PEUT fournir d'autres informations sur son état de disponibilité en utilisant l'élément <show/> (voir au paragraphe 2.2.2.1)).

Exemple : État de disponibilité :

```
<presence>
  <show>dnd</show>
</presence>
```

5.3 Spécification des informations détaillées d'état

En conjonction avec l'élément <show/>, un client PEUT fournir des informations d'état détaillées en utilisant l'élément <status/> (Voir au paragraphe 2.2.2.2 "État").

Exemple : Informations d'état détaillées :

```
<presence xml:lang='fr'>
  <show>dnd</show>
  <status>Courtiser Juliette</status>
  <status xml:lang='cz'>Ja dvo&#x0159;&#x00ED;m Juliet</status>
</presence>
```

5.4 Spécification de la priorité de présence

Un client PEUT fournir une priorité pour ses ressources en utilisant l'élément <priority/> (voir au paragraphe 2.2.2.3 "Priorité").

Exemple : Priorité de présence :

```
<presence xml:lang='en'>
  <show>dnd</show>
  <status>Wooing Juliet</status>
  <status xml:lang='cz'>Ja dvo&#x0159;&#x00ED;m Juliet</status>
  <priority>1</priority>
</presence>
```

5.5 Exemples de présence

Les exemples de ce paragraphe illustrent les protocoles en relations avec présence décrits ci-dessus. L'usager est romeo@exemple.net, il a une ressource disponible dont l'identifiant de ressource est "orchard", et il a les individus suivants sur son rôle :

- juliet@exemple.com (subscription="both" et elle a deux ressources disponibles, une dont la ressource est "chamber" et l'autre dont la ressource est "balcony")
- benvolio@exemple.org (subscription="to")
- mercutio@exemple.org (subscription="from")

Exemple 1 : L'usager envoie une présence initiale :

```
<presence/>
```

Exemple 2 : Le serveur de l'usager envoie des sondes de présences aux contacts qui ont subscription="to" et subscription="both" au nom de la ressource disponible de l'usager :

```
<presence
    type='probe'
    from='romeo@exemple.net/orchard'
    to='juliet@exemple.com'/>

<presence
    type='probe'
    from='romeo@exemple.net/orchard'
    to='benvolio@exemple.org'/>
```

Exemple 3 : Le serveur de l'usager envoie une présence initiale aux contacts qui ont subscription="from" et subscription="both" au nom des ressources disponibles de l'usager :

```
<presence
    from='romeo@exemple.net/orchard'
    to='juliet@exemple.com'/>

<presence
    from='romeo@exemple.net/orchard'
    to='mercutio@exemple.org'/>
```

Exemple 4 : Les serveurs des contacts répondent aux sondes de présence au nom de toutes les ressources disponibles :

```
<presence
    from='juliet@exemple.com/balcony'
    to='romeo@exemple.net/orchard'
    xml:lang='fr'>
  <show>away</show>
  <status>je reviens de suite</status>
  <priority>0</priority>
</presence>
```

```
<presence
    from='juliet@exemple.com/chamber'
    to='romeo@exemple.net/orchard'>
  <priority>1</priority>
</presence>

<presence
    from='benvolio@exemple.org/pda'
    to='romeo@exemple.net/orchard'
    xml:lang='en'>
  <show>dnd</show>
  <status>gallivanting</status>
</presence>
```

Exemple 5 : Les serveurs de contacts livrent la présence initiale de l'usager à toutes les ressources disponibles ou retournent une erreur à l'usager :

```
<presence
    from='romeo@exemple.net/orchard'
    to='juliet@exemple.com/chamber'/>

<presence
    from='romeo@exemple.net/orchard'
    to='juliet@exemple.com/balcony'/>

<presence
    type='error'
    from='mercutio@exemple.org'
    to='romeo@exemple.net/orchard'>
  <error type='cancel'>
    <gone xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</presence>
```

Exemple 6 : L'usager envoie une présence dirigée à un autre usager qui n'est pas dans son rôle :

```
<presence
    from='romeo@exemple.net/orchard'
    to='nurse@exemple.com'
    xml:lang='en'>
  <show>dnd</show>
  <status>courting Juliet</status>
  <priority>0</priority>
</presence>
```

Exemple 7 : L'usager envoie des informations de présence disponible mises à jour à diffuser :

```
<presence xml:lang='fr'>
  <show>away</show>
  <status>Je reviendrai !</status>
  <priority>1</priority>
</presence>
```

Exemple 8 : Le serveur de l'usager diffuse des informations de présence mises à jour à un seul contact (pas un de ceux d'où une erreur a été reçue ou à qui l'usager a envoyé une présence dirigée) :

```
<presence
    from='romeo@exemple.net/orchard'
    to='juliet@exemple.com'
    xml:lang='fr'>
  <show>away</show>
  <status>Je reviendrai !</status>
```

```
  <priority>1</priority>
</presence>
```

Exemple 9 : Le serveur du contact livre des informations de présence mises à jour à toutes les ressources disponibles du contact :

```
[à la ressource "balcony" ...]
<presence
    from='romeo@exemple.net/orchard'
    to='juliet@exemple.com'
    xml:lang='en'>
  <show>away</show>
  <status>I shall return!</status>
  <priority>1</priority>
</presence>

[à la ressource "chamber" ...]
<presence
    from='romeo@exemple.net/orchard'
    to='juliet@exemple.com'
    xml:lang='en'>
  <show>away</show>
  <status>I shall return!</status>
  <priority>1</priority>
</presence>
```

Exemple 10 : Une des ressources du contact diffuse une présence finale :

```
<presence from='juliet@exemple.com/balcony' type='unavailable'/>
```

Exemple 11 : Le serveur du contact envoie des informations de présence indisponible à l'usager :

```
<presence
    type='unavailable'
    from='juliet@exemple.com/balcony'
    to='romeo@exemple.net/orchard'/>
```

Exemple 12 : L'usager envoie une présence finale :

```
<presence from='romeo@exemple.net/orchard'
          type='unavailable'
          xml:lang='fr'>
  <status>rentré chez moi</status>
</presence>
```

Exemple 13 : Le serveur de l'usager diffuse des informations de présence indisponible au contact ainsi qu'à la personne à qui l'usager a envoyé la présence dirigée :

```
<presence
    type='unavailable'
    from='romeo@exemple.net/orchard'
    to='juliet@exemple.com'
    xml:lang='en'>
  <status>gone home</status>
</presence>

<presence
    from='romeo@exemple.net/orchard'
    to='nurse@exemple.com'
    xml:lang='en'>
  <status>gone home</status>
</presence>
```

6. Gestion des abonnements

Afin de protéger la confidentialité des utilisateurs de messagerie instantanée et de toutes les autres entités, les informations de présence et de disponibilité ne sont divulguées qu'aux autres entités que l'usager a approuvées. Lorsque un usager a donné son accord pour qu'une autre entité puisse voir sa présence, l'entité est dite avoir un abonnement aux informations de présence de l'usager. Un abonnement perdure à travers les sessions ; bien sûr, il dure jusqu'à ce que l'abonné se désabonne ou que l'entité objet de l'abonnement révoque l'abonnement accordé précédemment. Les abonnements sont gérés au sein de XMPP par l'envoi de strophes de présence contenant des attributs spécialement définis.

Note : Il y a des interactions importantes entre les abonnements et les rôles ; elles sont définies à la Section 8 "Intégration des éléments de rôle et des abonnements de présence", et le lecteur se réfèrera à cette section pour une compréhension complète des abonnements de présence.

6.1 Demande d'abonnement

Une demande d'abonnement à la présence d'une autre entité est faite par l'envoi d'une strophe de présence de type "subscribe".

Exemple : Envoi d'une demande d'abonnement :

<presence to='juliet@exemple.com' type='subscribe'/>

Pour les responsabilités du client et du serveur concernant les demandes d'abonnement de présence, se reporter au paragraphe 5.1.6 "Abonnements de présence".

6.2 Traitement d'une demande d'abonnement

Lorsque un client reçoit une demande d'abonnement d'une autre entité, il DOIT soit approuver la demande par l'envoi d'une strophe de présence de type "subscribed", soit refuser la demande par l'envoi d'une strophe de présence de type "unsubscribed".

Exemple : Approbation d'une demande d'abonnement :

<presence to='romeo@exemple.net' type='subscribed'/>

Exemple : Refus d'une demande d'abonnement de présence :

<presence to='romeo@exemple.net' type='unsubscribed'/>

6.3 Annulation d'un abonnement à une autre entité

Si un usager souhaite annuler une demande d'abonnement accordée précédemment, il envoie une strophe de présence de type "unsubscribed".

Exemple : Annulation d'une demande d'abonnement accordée précédemment :

<presence to='romeo@exemple.net' type='unsubscribed'/>

6.4 Désabonnement de la présence d'une autre entité

Si un usager souhaite se désabonner de la présence d'une autre entité, il envoie une strophe de présence de type "unsubscribe".

Exemple : Désabonnement de la présence d'une entité :

<presence to='juliet@exemple.com' type='unsubscribe'/>

7. Gestion du rôle

Dans XMPP, une liste de contacts est appelée un rôle, qui consiste en tout nombre d'éléments spécifiques du rôle, dont chacun est identifié par un JID unique (généralement de la forme <contact@domaine>). Le rôle d'un usager est mémorisé au nom de

l'usager par le serveur de l'usager afin que l'usager puisse accéder aux informations du rôle à partir de toute ressource.

Note : Il y a des interactions importantes entre les rôles et les abonnements ; elles sont définies à la Section 8 "Intégration des éléments de rôle et abonnements à présence", et le lecteur se reportera à cette section pour une compréhension complète de la gestion d'un rôle.

7.1 Syntaxe et sémantique

Les rôles sont gérés en utilisant les strophes IQ, spécifiquement au moyen d'un élément fils <query/> qualifié par l'espace de noms 'jabber:iq:roster'. L'élément <query/> PEUT contenir un ou plusieurs <item/> fils, dont chacun décrit un unique élément de rôle ou "contact".

La "clé" ou identifiant unique de chaque élément de rôle est un JID, encapsulé dans l'attribut 'jid' de l'élément <item/> (qui est EXIGÉ). La valeur de l'attribut 'jid' DEVRAIT être de la forme <usager@domaine> si l'élément est associé à un autre utilisateur (humain) de messagerie instantanée.

L'état de l'abonnement de présence en relation avec un élément de rôle est capturé dans l'attribut 'subscription' de l'élément <item/>. Les valeurs permises pour cet attribut sont :

"none" : l'usager n'a pas d'abonnement aux informations de présence du contact, et le contact n'a pas d'abonnement aux informations de présence de l'usager.

"to" : l'usager a un abonnement aux informations de présence du contact, mais le contact n'a pas d'abonnement aux informations de présence de l'usager.

"from" : le contact a un abonnement aux informations de présence de l'usager, mais l'usager n'a pas d'abonnement aux informations de présence du contact.

"both" : l'usager et le contact ont tous deux des abonnements aux informations de présence de l'autre.

Chaque élément <item/> PEUT contenir un attribut 'name', qui établit le "pseudonyme" *(nickname)* à associer au JID, comme déterminé par l'usager (et non par le contact). La valeur de l'attribut 'name' est opaque.

Chaque élément <item/> PEUT contenir un ou plusieurs éléments fils <group/>, à utiliser pour collecter des éléments du rôle en diverses catégories. Les données de caractères XML de l'élément <group/> sont opaques.

7.2 Règles de gestion

Un serveur DOIT ignorer toute adresse 'to' sur un rôle "set", et DOIT traiter tout rôle "set" comme appliqué à l'envoyeur. Pour augmenter la sécurité, un client DEVRAIT vérifier l'adresse "from" d'un "rôle poussé" (IQ entrante et type "set" contenant un élément de rôle) pour s'assurer qu'il vient d'une source de confiance ; précisément, la strophe DOIT soit ne pas avoir d'attribut 'from' (c'est-à-dire, implicitement du serveur) soit avoir un attribut 'from' dont la valeur correspond au JID nu de l'usager (de forme <usager@domaine>) ou au JID complet (de forme <usager@domaine/ressource>) ; autrement, le client DEVRAIT ignorer les "rôles poussés".

7.3 Restitution d'un rôle à l'amorçage

À la connexion au serveur et en devenant une ressource active, un client DEVRAIT demander le rôle avant d'envoyer une présence initiale (cependant, parce qu'il peut n'être pas souhaitable de recevoir le rôle pour toutes les ressources, par exemple, une connexion avec une bande passante limitée, la demande du rôle par le client est FACULTATIVE). Si une ressource disponible ne demande pas le rôle durant une session, le serveur NE DOIT PAS lui envoyer d'abonnement de présence et les mises à jour du rôle associées.

Exemple : Le client demande au serveur le rôle actuel :

```
<iq from='juliet@exemple.com/balcony' type='get' id='roster_1'>
  <query xmlns='jabber:iq:roster'/>
</iq>
```

Exemple : Le client reçoit le rôle du serveur :

```
<iq to='juliet@exemple.com/balcony' type='result' id='roster_1'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@exemple.net'
          name='Romeo'
          subscription='both'>
      <group>Friends</group>
    </item>
    <item jid='mercutio@exemple.org'
          name='Mercutio'
          subscription='from'>
      <group>Friends</group>
    </item>
    <item jid='benvolio@exemple.org'
          name='Benvolio'
          subscription='both'>
      <group>Friends</group>
    </item>
  </query>
</iq>
```

7.4 Ajout d'un élément au rôle

À tout moment, un usager PEUT ajouter un élément à son rôle.

Exemple : Le client ajoute un nouvel élément :

```
<iq from='juliet@exemple.com/balcony' type='set' id='roster_2'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@exemple.com'
          name='Nurse'>
      <group>Servants</group>
    </item>
  </query>
  </iq>
```

Le serveur DOIT mettre à jour les informations du rôle dans une mémorisation persistante, et aussi pousser les changements à toutes les ressources disponibles de l'usager qui ont demandé le rôle. Cette "poussée de rôle" consiste en une strophe IQ de type "set" du serveur au client et qui permet à toutes les ressources disponibles de rester synchronisées avec les informations de rôle fondées sur le serveur.

Exemple : Le serveur (1) pousse les informations de rôle mises à jour à toutes les ressources disponibles qui ont demandé le rôle et (2) répond avec un résultat IQ à la ressource envoyeuse :

```
<iq to='juliet@exemple.com/balcony'
    type='set'
    id='a78b4q6ha463'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@exemple.com'
          name='Nurse'
          subscription='none'>
      <group>Servants</group>
    </item>
  </query>
</iq>

<iq to='juliet@exemple.com/chamber'
    type='set'
    id='a78b4q6ha464'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@exemple.com'
          name='Nurse'
          subscription='none'>
```

```
    <group>Servants</group>
   </item>
  </query>
</iq>

<iq to='juliet@exemple.com/balcony' type='result' id='roster_2'/>
```

La sémantique de strophe de type IQ exige, comme défini dans la [RFC3920], que chaque ressource qui a reçu le rôle poussé DOIT répondre avec une strophe IQ de type "result" (ou "error").

Exemple : Les ressources répondent par une IQ de résultat au serveur :

```
<iq from='juliet@exemple.com/balcony'
   to='exemple.com'
   type='result'
   id='a78b4q6ha463'/>
<iq from='juliet@exemple.com/chamber'
   to='exemple.com'
   type='result'
   id='a78b4q6ha464'/>
```

7.5 Mise à jour d'un élément du rôle

La mise à jour d'un élément de rôle existant (par exemple, changer le groupe) se fait de la même façon que l'ajout d'un nouvel élément de rôle, c'est-à-dire, en envoyant l'élément de rôle dans une IQ établie au serveur.

Exemple : L'usager met à jour un élément du rôle (ajout d'un groupe) :

```
<iq from='juliet@exemple.com/chamber' type='set' id='roster_3'>
  <query xmlns='jabber:iq:roster'>
    <item jid='romeo@exemple.net'
          name='Romeo'
          subscription='both'>
      <group>Friends</group>
      <group>Lovers</group>
    </item>
  </query>
</iq>
```

Comme pour l'ajout d'un élément du rôle, lorsque le serveur met à jour un élément du rôle, il DOIT mettre à jour les informations du rôle dans une mémorisation persistante, et aussi initier une poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle.

7.6 Suppression d'un élément du rôle

À tout moment, un usager PEUT supprimer un élément de son rôle par l'envoi d'une IQ établie au serveur et en s'assurant que la valeur de l'attribut 'subscription' est "remove" (un serveur conforme DOIT ignorer toutes les autres valeurs de l'attribut 'subscription' reçues d'un client).

Exemple : Le client supprime un élément :

```
<iq from='juliet@exemple.com/balcony' type='set' id='roster_4'>
  <query xmlns='jabber:iq:roster'>
    <item jid='nurse@exemple.com' subscription='remove'/>
  </query>
</iq>
```

Comme avec l'ajout d'un élément de rôle, lorsque le serveur supprime un élément du rôle, il DOIT mettre à jour les informations du rôle dans une mémorisation persistante, initier une poussée du rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle (avec l'attribut 'subscription' réglé à une valeur de "remove") et envoyer une IQ de 'result' à la ressource initiatrice.

Pour plus d'informations sur les implications de cette commande, voir le paragraphe 8.6 "Suppression d'un élément de rôle et annulation de tous les abonnements".

8. Intégration d'éléments de rôle et abonnements à présence

8.1 Vue d'ensemble

On s'attend à un certain niveau d'intégration entre les éléments de rôle et les abonnements de présence chez un usager de messagerie instantanée en ce qui concerne les abonnements de l'usager à d'autres contacts, et de la part de ceux-ci. La présente section décrit le niveau d'intégration qui DOIT être accepté aux sein des applications de messagerie instantanée XMPP.

Il y a quatre états d'abonnement principaux :

None : l'usager n'a pas d'abonnement aux informations de présence du contact, et le contact n'a pas d'abonnement aux informations de présence de l'usager.

To : l'usager a un abonnement aux informations de présence du contact, mais le contact n'a pas d'abonnement aux informations de présence de l'usager.

From : le contact a un abonnement aux informations de présence de l'usager, mais l'usager n'a pas d'abonnement aux informations de présence du contact.

Both : l'usager et le contact ont tous deux un abonnement aux informations de présence de l'autre (c'est-à-dire, l'union de 'from' et de 'to')

Chacun de ces états se reflète dans le rôle de l'usager et du contact, résultant donc en états d'abonnement durables.

Les explications de la façon dont ces états d'abonnement interagissent avec les éléments de rôle afin de réaliser certains cas d'utilisation définis sont fournies dans les paragraphes qui suivent. Tous les détails concernant le traitement par le serveur et le client de tous les états d'abonnement (incluant les états en cours entre les états principaux énumérés ci-dessus) sont fournis à la Section 9 "États d'abonnement".

Le serveur NE DOIT PAS envoyer de demande d'abonnement de présence ou de poussées de rôle aux ressources indisponibles, ni aux ressources disponibles qui n'ont pas demandé le rôle.

Les adresses 'from' et 'to' sont FACULTATIVES dans les poussées de rôle ; si elles sont incluses, leur valeur DEVRAIT être le JID complet de la ressource pour cette session. Un client DOIT accuser réception d'une poussée de rôle par une strophe IQ de type "result" (pour faire court, ces strophes ne sont pas montrées dans les exemples qui suivent mais elles sont exigées par la sémantique de IQ définie dans la [RFC3920]).

8.2 Abonnement d'un usager à Contact

Le processus par lequel un usager s'abonne à un contact, incluant l'interaction entre les éléments de rôle et les états d'abonnement, est décrit ci-dessous :

1. Pour se préparer à être capable de rendre le contact dans l'interface de client de l'usager et pour que le serveur garde trace de l'abonnement, le client de l'usager DEVRAIT effectuer un "établissement de rôle" pour le nouvel élément du rôle. Cette demande consiste en l'envoi d'une strophe IQ de type='set' contenant un élément <query/> qualifié par l'espace de noms 'jabber:iq:roster', qui à son tour contient un élément <item/> qui définit le nouvel élément du rôle ; l'élément <item/> DOIT posséder un attribut 'jid', PEUT posséder un attribut 'name', NE DOIT PAS posséder un attribut 'subscription', et PEUT contenir un ou plusieurs éléments fils <group/>:

```
<iq type='set' id='set1'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='contact@exemple.org'
       name='MyContact'>
     <group>MyBuddies</group>
    </item>
  </query>
```

```
</iq>
```

2. Par suite, le serveur de l'usager (1) DOIT initier une poussée de rôle pour le nouvel élément de rôle pour toutes les ressources disponibles associées à cet usager qui ont demandé le rôle, en réglant l'attribut 'subscription' à une valeur de "none"; et (2) DOIT répondre à la ressource envoyeuse par un résultat IQ qui indique le succès de l'établissement du rôle :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='contact@exemple.org'
       subscription='none'
       name='MyContact'>
      <group>MyBuddies</group>
    </item>
  </query>
</iq>

<iq type='result' id='set1'/>
```

3. Si l'usager veut demander un abonnement aux informations de présence du contact, le client de l'usager DOIT envoyer une strophe de présence de type='subscribe' au contact :

```
<presence to='contact@exemple.org' type='subscribe'/>
```

4. Par suite, le serveur de l'usager DOIT initier une seconde poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle, en réglant le contact au sous état 'pending' de l'état d'abonnement de 'none' ; ce sous état de 'pending' est noté par l'inclusion de l'attribut ask='subscribe' dans l'élément du rôle :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='contact@exemple.org'
       subscription='none'
       ask='subscribe'
       name='MyContact'>
      <group>MyBuddies</group>
    </item>
  </query>
</iq>
```

Note : Si l'usager n'a pas créé un élément de rôle avant d'envoyer la demande d'abonnement, le serveur DOIT maintenant en créer un au nom de l'usager, puis envoyer une poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle, en omettant l'attribut 'name' et le <group/> fils montré ci-dessus.

5. Le serveur de l'usager DOIT aussi marquer la strophe de présence de type "subscribe" avec le JID nu de l'usager (c'est-à-dire, <usager@exemple.com>) comme l'adresse 'from' (si l'usager a fourni une adresse 'from' réglée au JID complet de l'usager, le serveur DEVRAIT retirer l'identifiant de ressource). Si le contact est desservi par un hôte différent de celui de l'usager, le serveur de l'usager DOIT acheminer la strophe de présence au serveur du contact pour livraison au contact (ce cas est supposé partout ; cependant, si le contact est desservi par le même hôte, le serveur peut alors simplement livrer directement la strophe de présence) :

```
<presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='subscribe'/>
```

Note : Si le serveur de l'usager reçoit une strophe de présence de type "error" du serveur du contact, il DOIT livrer la strophe d'erreur à l'usager, dont le client PEUT déterminer que l'erreur est en réponse à la strophe de présence sortante de type "subscribe" qu'il a envoyée précédemment (par exemple, en retraçant un attribut 'id') et ensuite choisir de renvoyer la demande "subscribe" ou faire revenir le rôle à son état précédent en envoyant une strophe de présence de type "unsubscribe" au contact.

6. À réception de la strophe de présence de type "subscribe" adressée au contact, le serveur du contact DOIT déterminer si il y

a au moins une ressource disponible à partir de laquelle le contact a demandé le rôle. Si il en est une, il DOIT livrer la demande d'abonnement au contact (sinon, le serveur du contact DOIT mémoriser hors ligne la demande d'abonnement pour la livrer lorsque cette condition sera satisfaite ; normalement ceci se fait en ajoutant un élément de rôle pour le contact au rôle de l'usager, avec un état de "None + Pending In" comme défini à la Section 9 "États d'abonnement", cependant un serveur NE DEVRAIT PAS pousser ou livrer des éléments de rôle dans cet état au contact). Sans égard au moment où la demande d'abonnement est livrée, le contact doit décider si il l'approuve ou non (sous réserve des préférences configurées du contact, le client du contact PEUT approuver ou refuser la demande d'abonnement sans la présenter au contact). On suppose ici la "voie heureuse" où le contact approuve la demande d'abonnement (l'autre flux de refus de la demande d'abonnement est défini au paragraphe 8.2.1). Dans ce cas, le client du contact (1) DEVRAIT effectuer un établissement de rôle spécifiant le pseudonyme et le groupe désiré pour l'usager (si il en est) ; et (2) DOIT envoyer une strophe de présence de type "subscribed" à l'usager afin d'approuver la demande d'abonnement.

```
<iq type='set' id='set2'>
  <query xmlns='jabber:iq:roster'>
    <item
        jid='user@exemple.com'
        name='SomeUser'>
      <group>SomeGroup</group>
    </item>
  </query>
</iq>

<presence to='user@exemple.com' type='subscribed'/>
```

7. Par suite, le serveur du contact (1) DOIT initier une poussée de rôle à toutes les ressources disponibles associées au contact qui ont demandé le rôle, contenant un élément de rôle pour l'usager avec l'état d'abonnement réglé à 'from' (le serveur DOIT l'envoyer même si le contact n'a pas effectué d'établissement de rôle) ; (2) DOIT retourner un résultat IQ à la ressource envoyeuse indiquant le succès de l'établissement du rôle ; (3) DOIT acheminer une strophe de présence de type "subscribed" à l'usager, en marquant d'abord l'adresse 'from' comme JID nu (<contact@exemple.org>) du contact; et (4) DOIT envoyer une présence disponible à partir de toutes les ressources disponibles du contact à l'usager :

```
<iq type='set' to='contact@exemple.org/resource'>
  <query xmlns='jabber:iq:roster'>
    <item
        jid='user@exemple.com'
        subscription='from'
        name='SomeUser'>
      <group>SomeGroup</group>
    </item>
  </query>
</iq>

<iq type='result' to='contact@exemple.org/resource' id='set2'/>

<presence
    from='contact@exemple.org'
    to='user@exemple.com'
    type='subscribed'/>

<presence
    from='contact@exemple.org/resource'
    to='user@exemple.com'/>
```

Note : Si le serveur du contact reçoit une strophe de présence de type "error" du serveur de l'usager, il DOIT livrer la strophe d'erreur au contact, dont le client PEUT déterminer que l'erreur est en réponse à la strophe de présence sortante de type "subscribed" qu'il a envoyée précédemment (par exemple, en retraçant un attribut 'id') et ensuite choisir de renvoyer la notification "subscribed" ou faire revenir le rôle à son état antérieur par l'envoi d'une strophe de présence de type "unsubscribed" à l'usager.

8. À réception de la strophe de présence de type "subscribed" adressée à l'usager, le serveur de l'usager DOIT d'abord vérifier que le contact est dans le rôle de l'usager avec l'un des états suivants : (a) subscription='none' et ask='subscribe' ou (b) subscription='from' et ask='subscribe'. Si le contact n'est pas sur le rôle de l'usager avec un de ces états, le serveur de l'usager DOIT ignorer en silence la strophe de présence de type "subscribed" (c'est-à-dire, il NE DOIT PAS l'acheminer à

l'usager, modifier le rôle de l'usager, ou générer une poussée de rôle aux ressources disponibles de l'usager). Si le contact est sur le rôle de l'usager avec un de ces états, le serveur de l'usager (1) DOIT livrer la strophe de présence de type "subscribed" provenant du contact à l'usager, (2) DOIT initier une poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle, contenant un élément de rôle mis à jour pour le contact avec l'attribut 'subscription' réglé à une valeur de to", et (3) DOIT livrer la strophe de présence disponible reçue de chacune des ressources disponibles du contact à chacune des ressources disponibles de l'usager :

```
<presence
    to='user@exemple.com'
    from='contact@exemple.org'
    type='subscribed'/>

<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
        jid='contact@exemple.org'
        subscription='to'
        name='MyContact'>
      <group>MyBuddies</group>
    </item>
  </query>
</iq>

<presence
    from='contact@exemple.org/resource'
    to='user@exemple.com/resource'/>
```

9. À réception de la strophe de présence de type "subscribed", l'usager DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "subscribe" au contact, soit en la "refusant" par l'envoi d'une strophe de présence de type "unsubscribe" au contact ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais laisse plutôt savoir au serveur de l'usager qu'il NE DOIT PLUS envoyer de notification du changement de l'état d'abonnement à l'usager (voir au paragraphe 9.4).

Du point de vue de l'usager, il existe maintenant un abonnement aux informations de présence du contact ; du point de vue du contact, il existe maintenant un abonnement de l'usager.

8.2.1 Autre flux : le contact refuse la demande d'abonnement

Le flux d'activité ci-dessus présente le "chemin heureux" de la demande d'abonnement de l'usager au contact. L'autre flux principal se présente si le contact refuse la demande d'abonnement de l'usager, comme décrit ci-dessous.

1. Si le contact veut refuser la demande, le client du contact DOIT envoyer une strophe de présence du type "unsubscribed" à l'usager (au lieu d'une strophe de présence du type "subscribed" envoyée à l'étape 6 du paragraphe 8.2) :

```
<presence to='usager@exemple.com' type='unsubscribed'/>
```

2. Par suite, le serveur du contact DOIT acheminer la strophe de présence de type "unsubscribed" à l'usager, en marquant d'abord l'adresse 'from' comme le JID nu (<contact@exemple.org>) du contact :

```
<presence
    from='contact@exemple.org'
    to='user@exemple.com'
    type='unsubscribed'/>
```

Note : Si le serveur du contact avait précédemment ajouté l'usager au rôle du contact pour des besoins de traçage, il DOIT retirer l'élément pertinent à ce moment.

3. À réception de la strophe de présence de type "unsubscribed" adressée à l'usager, le serveur de l'usager (1) DOIT livrer cette strophe de présence à l'usager et (2) DOIT initier une poussée de rôle à toutes les ressources disponibles de l'usager qui avaient demandé le rôle, contenant un élément de rôle mis à jour pour le contact avec l'attribut 'subscription' réglé à une valeur de "none" et sans attribut 'ask' :

```
<presence
   from='contact@exemple.org'
   to='user@exemple.com'
   type='unsubscribed'/>

<iq type='set'>
 <query xmlns='jabber:iq:roster'>
  <item
     jid='contact@exemple.org'
     subscription='none'
     name='MyContact'>
    <group>MyBuddies</group>
  </item>
 </query>
</iq>
```

4. À réception de la strophe de présence de type "unsubscribed", l'usager DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "unsubscribe" au contact, soit en la "refusant" par l'envoi d'une strophe de présence de type "subscribe" au contact ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur de l'usager qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement à l'usager (voir le paragraphe 9.4).

Le résultat de cette activité est que le contact est maintenant sur le rôle de l'usager avec un état d'abonnement de "none", tandis que l'usager n'est plus du tout sur le rôle du contact.

8.3 Création d'un abonnement mutuel

L'usager et le contact peuvent s'appuyer sur le "chemin heureux" décrit ci-dessus pour créer un abonnement mutuel (c'est-à-dire, un abonnement de type "both"). Le processus est décrit ci-dessous.

1. Si le contact veut créer un abonnement mutuel, le contact DOIT envoyer une demande d'abonnement à l'usager (sous réserve des préférences configurées du contact, le client du contact PEUT l'envoyer automatiquement) :

```
<presence to='user@exemple.com' type='subscribe'/>
```

2. Par suite, le serveur du contact (1) DOIT initier une poussée de rôle à toutes les ressources disponibles associées au contact qui ont demandé le rôle, avec l'usager encore dans l'état d'abonnement 'from' mais avec un abonnement 'to' en cours noté par l'inclusion de l'attribut ask='subscribe' dans l'élément du rôle ; et (2) DOIT acheminer la strophe de présence de type "subscribe" à l'usager, en marquant d'abord l'adresse 'from' comme JID nu (<contact@exemple.org>) du contact :

```
<iq type='set'>
 <query xmlns='jabber:iq:roster'>
  <item
     jid='user@exemple.com'
     subscription='from'
     ask='subscribe'
     name='SomeUser'>
    <group>SomeGroup</group>
  </item>
 </query>
</iq>

<presence
   from='contact@exemple.org'
   to='user@exemple.com'
   type='subscribe'/>
```

Note : Si le serveur du contact reçoit une strophe de présence de type "error" du serveur de l'usager, il DOIT livrer la strophe d'erreur au contact, dont le client PEUT déterminer que l'erreur est en réponse à la strophe de présence sortante de type "subscribe" qu'il a envoyée précédemment (par exemple, en retraçant un attribut 'id') et ensuite choisir de renvoyer la demande "subscribe" ou de revenir à l'état antérieur du rôle par l'envoi d'une strophe de présence de type "unsubscribe" à l'usager.

3. À réception de la strophe de présence de type "subscribe" adressée à l'usager, le serveur de l'usager doit déterminer si il y a au moins une ressource disponible pour laquelle l'usager a demandé le rôle. Si il y en a une, le serveur de l'usager DOIT livrer la demande d'abonnement à l'usager (sinon, il DOIT mémoriser la demande d'abonnement hors ligne pour la livrer quand cette condition sera satisfaite). Peu importe que la demande d'abonnement soit livrée, l'usager doit alors décider si il l'approuve ou non (sous réserve des préférences configurées de l'usager, le client de l'usager PEUT approuver ou refuser la demande d'abonnement sans la présenter à l'usager). On suppose ici le "chemin heureux" où l'usager approuve la demande d'abonnement (l'autre flux où il refuse la demande d'abonnement est défini au paragraphe 8.3.1). Dans ce cas, le client de l'usager DOIT envoyer une strophe de présence de type "subscribed" au contact afin d'approuver la demande d'abonnement.

```
<presence to='contact@exemple.org' type='subscribed'/>
```

4. Par suite, le serveur de l'usager (1) DOIT initier une poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle, contenant un élément de rôle pour le contact avec l'attribut 'subscription' réglé à une valeur de "both" ; (2) DOIT acheminer la strophe de présence de type "subscribed" au contact, en marquant d'abord l'adresse 'from' comme le JID nu (<user@exemple.com>) de l'usager ; et (3) DOIT envoyer au contact le XML complet de la dernière strophe de présence sans l'attribut 'to' reçue par le serveur de chaque ressource disponible de l'usager (sous réserve des listes de confidentialité en vigueur pour chaque session) :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='contact@exemple.org'
       subscription='both'
       name='MyContact'>
      <group>MyBuddies</group>
    </item>
  </query>
</iq>

<presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='subscribed'/>

<presence
    from='user@exemple.com/resource'
    to='contact@exemple.org'/>
```

Note : Si le serveur de l'usager reçoit une strophe de présence de type "error" du serveur du contact, il DOIT livrer la strophe d'erreur à l'usager, dont le client PEUT déterminer que l'erreur est en réponse à la strophe de présence sortante de type "subscribed" qu'il a envoyée antérieurement (par exemple, an retraçant un attribut 'id') et ensuite choisir de renvoyer la demande d'abonnement ou faire revenir le rôle à son état antérieur par l'envoi d'une strophe de présence de type "unsubscribed" au contact.

5. À réception de la strophe de présence de type "subscribed" adressée au contact, le serveur du contact DOIT d'abord vérifier que l'usager est sur le rôle du contact avec l'un des états suivants : (a) subscription='none' et ask='subscribe' ou (b) subscription='from' et ask='subscribe'. Si l'usager n'est pas sur le rôle du contact avec un de ces états, le serveur du contact DOIT ignorer en silence la strophe de présence de type "subscribed" (c'est-à-dire, il NE DOIT PAS l'acheminer au contact, modifier le rôle du contact, ou générer une poussée de rôle aux ressources disponibles du contact). Si l'usager est sur le rôle du contact avec un de ces états, le serveur du contact (1) DOIT livrer la strophe de présence de type "subscribed" provenant de l'usager au contact, (2) DOIT initier une poussée de rôle à toutes les ressources disponibles associées au contact qui ont demandé le rôle, contenant un élément mis à jour du rôle pour l'usager avec l'attribut 'subscription' réglé à une valeur de "both", et (3) DOIT livrer la strophe de présence disponible reçue de chaque ressource disponible de l'usager à chaque ressource disponible du contact :

```
<presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='subscribed'/>

<iq type='set'>
  <query xmlns='jabber:iq:roster'>
```

```
    <item
       jid='user@exemple.com'
       subscription='both'
       name='SomeUser'>
      <group>SomeGroup</group>
    </item>
  </query>
 </iq>

 <presence
    from='user@exemple.com/resource'
    to='contact@exemple.org/resource'/>
```

6. À réception de la strophe de présence de type "subscribed", le contact DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "subscribe" à l'usager, soit en la "refusant" par l'envoi d'une strophe de présence de type "unsubscribe" à l'usager ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur du contact qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement au contact (voir le paragraphe 9.4).

L'usager et le contact ont maintenant un abonnement mutuel à la présence de l'autre -- c'est-à-dire, l'abonnement est du type "both".

8.3.1 Autre flux : l'usager refuse la demande d'abonnement

Le flux d'activité ci-dessus présente le "chemin heureux" du la demande d'abonnement à l'usager du contact. Le principal autre flux survient si l'usager refuse la demande d'abonnement du contact, comme décrit ci-dessous :

1. Si l'usager veut refuser la demande, le client de l'usager DOIT envoyer une strophe de présence de type "unsubscribed" au contact (au lieu de la strophe de présence de type "subscribed" envoyée à l'étape 3 du paragraphe 8.3):

```
  <presence to='contact@exemple.org' type='unsubscribed'/>
```

2. Par suite, le serveur de l'usager DOIT acheminer la strophe de présence de type "unsubscribed" au contact, en marquant d'abord l'adresse 'from' comme JID nu (<user@exemple.com>) de l'usager :

```
 <presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='unsubscribed'/>
```

3. À réception de la strophe de présence de type "unsubscribed" adressée au contact, le serveur du contact (1) DOIT livrer cette strophe de présence au contact, et (2) DOIT initier une poussée de rôle à toutes les ressource disponibles associées au contact qui ont demandé le rôle, contenant un élément de rôle mis à jour pour l'usager avec l'attribut 'subscription' réglé à une valeur de "from" et sans attribut 'ask' :

```
 <presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='unsubscribed'/>

 <iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='user@exemple.com'
       subscription='from'
       name='SomeUser'>
      <group>SomeGroup</group>
    </item>
  </query>
 </iq>
```

4. À réception de la strophe de présence de type "unsubscribed", le contact DEVRAIT accuser réception de cette notification

d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "unsubscribe" à l'usager, soit en la "refusant" par l'envoi d'une strophe de présence de type "subscribe" à l'usager ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur du contact qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement au contact (voir le paragraphe 9.4).

Par suite de cette activité, il n'y a pas eu de changement de l'état d'abonnement, c'est-à-dire, le contact est dans le rôle de l'usager avec un état d'abonnement de "to" et l'usager est dans le rôle du contact avec un état d'abonnement de "from".

8.4 Désabonnement

À tout moment après l'abonnement aux informations de présence d'un contact, un usager PEUT se désabonner. Bien que le XML qu'envoie l'usager pour provoquer cela soit le même dans toutes les instances, l'état d'abonnement qui suit est différent selon l'état d'abonnement obtenu lors de l'envoi de la "commande" unsubscribe. Les deux scénarios possibles sont décrits ci-dessous.

8.4.1 Cas n° 1 : désabonnement lorsque l'abonnement n'est pas mutuel

Dans le premier cas, l'usager a un abonnement aux informations de présence du contact mais le contact n'a pas d'abonnement aux informations de présence de l'usager (c'est-à-dire, l'abonnement n'est pas encore mutuel).

1. Si l'usager veut se désabonner des informations de présence du contact, il DOIT envoyer une strophe de présence de type "unsubscribe" au contact :

```
<presence to='contact@exemple.org' type='unsubscribe'/>
```

2. Par suite, le serveur de l'usager (1) DOIT envoyer une poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle, contenant un élément de rôle mis à jour pour le contact avec l'attribut 'subscription' réglé à une valeur de "none"; et (2) DOIT acheminer la strophe de présence de type "unsubscribe" au contact, en marquant d'abord l'adresse 'from' comme JID nu (<usager@exemple.com>) de l'usager :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='contact@exemple.org'
       subscription='none'
       name='MyContact'>
      <group>MyBuddies</group>
    </item>
  </query>
</iq>

<presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='unsubscribe'/>
```

3. À réception de la strophe de présence de type "unsubscribe" adressée au contact, le serveur du contact (1) DOIT initier une poussée de rôle à toutes les ressources disponibles associées au contact qui ont demandé le rôle, contenant un élément de rôle mis à jour pour l'usager avec l'attribut 'subscription' réglé à une valeur de "none" (si le contact est indisponible ou n'a pas demandé le rôle, le serveur du contact DOIT modifier l'élément de rôle et envoyer cet élément modifié la prochaine fois que le contact demandera le rôle) et (2) DOIT livrer la notification du changement de l'état "unsubscribe" au contact :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='user@exemple.com'
       subscription='none'
       name='SomeUser'>
      <group>SomeGroup</group>
    </item>   </query>
</iq>
```

```
<presence
  from='user@exemple.com'
  to='contact@exemple.org'
  type='unsubscribe'/>
```

4. À réception de la strophe de présence de type "unsubscribe", le contact DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "unsubscribed" à l'usager, soit en la "refusant" par l'envoi d'une strophe de présence de type "subscribed" à l'usager ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur du contact qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement au contact (voir le paragraphe 9.4).

5. Le serveur du contact (1) DOIT alors envoyer une strophe de présence de type "unsubscribed" à l'usager, et (2) DEVRAIT envoyer à l'usager une présence indisponible à partir de toutes les ressource disponibles du contact :

```
<presence
  from='contact@exemple.org'
  to='user@exemple.com'
  type='unsubscribed'/>

<presence
  from='contact@exemple.org/resource'
  to='user@exemple.com'
  type='unavailable'/>
```

6. Lorsque le serveur de l'usager reçoit la strophe de présences de type "unsubscribed" et "unavailable", il DOIT les livrer à l'usager :

```
<presence
  from='contact@exemple.org'
  to='user@exemple.com'
  type='unsubscribed'/>

<presence
  from='contact@exemple.org/resource'
  to='user@exemple.com'
  type='unavailable'/>
```

7. À réception de la strophe de présence de type "unsubscribed", l'usager DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "unsubscribe" au contact, soit en la "refusant" par l'envoi d'une strophe de présence de type "subscribe" au contact ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur de l'usager qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement à l'usager (voir le paragraphe 9.4).

8.4.2 Cas n° 2 : désabonnement lorsque l'abonnement est mutuel

Dans le second cas, l'usager a un abonnement aux informations de présence du contact et le contact a aussi un abonnement aux informations de présence de l'usager (c'est-à-dire, l'abonnement est mutuel).

1. Si l'usager veut se désabonner des informations de présence du contact, l'usager DOIT envoyer une strophe de présence de type "unsubscribe" au contact :

```
<presence to='contact@exemple.org' type='unsubscribe'/>
```

2. Par suite, le serveur de l'usager (1) DOIT envoyer une poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle, contenant un élément de rôle mis à jour pour le contact avec l'attribut 'subscription' réglé à la valeur de "from" ; et (2) DOIT acheminer la strophe de présence de type "unsubscribe" au contact, en marquant d'abord l'adresse 'from' comme JID nu (<usager@exemple.com>) de l'usager :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
      jid='contact@exemple.org'
```

```
      subscription='from'
      name='MyContact'>
    <group>MyBuddies</group>
   </item>
  </query>
</iq>

<presence
   from='user@exemple.com'
   to='contact@exemple.org'
   type='unsubscribe'/>
```

3. À réception de la strophe de présence de type "unsubscribe" adressée au contact, le serveur du contact (1) DOIT initier une poussée de rôle à toutes les ressources disponibles associées au contact qui ont demandé le rôle, contenant un élément de rôle mis à jour pour l'usager avec l'attribut 'subscription' réglé à la valeur de "to" (si le contact est indisponible ou n'a pas demandé le rôle, le serveur du contact DOIT modifier l'élément de rôle et envoyer cet élément modifié la prochaine fois que le contact demande le rôle) ; et (2) DOIT livrer la notification de changement d'état "unsubscribe" au contact :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
      jid='user@exemple.com'
      subscription='to'
      name='SomeUser'>
     <group>SomeGroup</group>
    </item>
  </query>
</iq>

<presence
   from='user@exemple.com'
   to='contact@exemple.org'
   type='unsubscribe'/>
```

4. À réception de la strophe de présence de type "unsubscribe", le contact DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "unsubscribed" à l'usager, soit en la "refusant" par l'envoi d'une strophe de présence de type "subscribed" à l'usager ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur du contact qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement au contact (voir le paragraphe 9.4).

5. Le serveur du contact (1) DOIT alors envoyer une strophe de présence de type "unsubscribed" à l'usager, et (2) DEVRAIT envoyer une présence indisponible à partir de toutes les ressources disponibles du contact à l'usager:

```
<presence
   from='contact@exemple.org'
   to='user@exemple.com'
   type='unsubscribed'/>

<presence
   from='contact@exemple.org/resource'
   to='user@exemple.com'
   type='unavailable'/>
```

6. Lorsque le serveur de l'usager reçoit les strophes de présence de type"unsubscribed" et "unavailable", il DOIT les livrer à l'usager :

```
<presence
   from='contact@exemple.org'
   to='user@exemple.com'
   type='unsubscribed'/>

<presence
   from='contact@exemple.org/resource'
```

```
    to='user@exemple.com'
    type='unavailable'/>
```

7. À réception de la strophe de présence de type "unsubscribed", l'usager DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "unsubscribe" au contact, soit en la "refusant" par l'envoi d'une strophe de présence de type "subscribe" au contact ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur de l'usager qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement à l'usager (voir le paragraphe 9.4).

Note : Évidemment il ne résulte pas de cela la suppression de l'élément de rôle du rôle de l'usager, et le contact a toujours son abonnement aux informations de présence de l'usager. Afin d'annuler complètement un abonnement mutuel et retirer pleinement l'élément de rôle du rôle de l'usager, l'usager DEVRAIT mettre à jour l'élément de rôle avec subscription='remove' comme défini au paragraphe 8.6 "Suppression d'un élément de rôle et annulation de tous les abonnements".

8.5 Annulation d'un abonnement

À tout moment après l'approbation d'une demande d'abonnement de la part d'un usager, un contact PEUT annuler cet abonnement. Bien que le XML qu'envoie le contact pour effectuer cela soit le même dans toutes les instances, l'état d'abonnement qui suit est différent selon l'état d'abonnement obtenu lorsque l'annulation est envoyée. Les deux scénarios possibles sont décrits ci-dessous.

8.5.1 Cas n°1 : annulation lorsque l'abonnement n'est pas mutuel

Dans le premier cas, l'usager a un abonnement aux informations de présence du contact mais le contact n'a pas d'abonnement aux informations de présence de l'usager (c'est-à-dire, l'abonnement n'est pas encore mutuel).

1. Si le contact veut annuler l'abonnement de l'usager, il DOIT envoyer une strophe de présence de type "unsubscribed" à l'usager :

```
<presence to='usager@exemple.com' type='unsubscribed'/>
```

2 Par suite, le serveur du contact (1) DOIT envoyer une poussée de rôle à toutes les ressources disponibles du contact qui ont demandé le rôle, contenant un élément de rôle mis à jour pour l'usager avec l'attribut 'subscription' réglé à une valeur de "none", (2) DOIT acheminer la strophe de présence de type "unsubscribed" à l'usager, en marquant d'abord l'adresse 'from' comme JID nu (<contact@exemple.org>) du contact, et (3) DEVRAIT envoyer une présence indisponible à partir de toutes les ressources disponibles du contact à l'usager :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='user@exemple.com'
       subscription='none'
       name='SomeUser'>
      <group>SomeGroup</group>
    </item>
  </query>
</iq>

<presence
    from='contact@exemple.org'
    to='user@exemple.com'
    type='unsubscribed'/>

<presence
    from='contact@exemple.org/resource'
    to='user@exemple.com'
    type='unavailable'/>
```

3. À réception de la strophe de présence de type "unsubscribed" adressée à l'usager, le serveur de l'usager (1) DOIT initier une poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle, contenant un élément de rôle mis à jour pour le contact avec l'attribut 'subscription' réglé à la valeur de "none" (si l'usager est indisponible ou n'a pas demandé

le rôle, le serveur de l'usager DOIT modifier l'élément de rôle et envoyer cet élément modifié la prochaine fois que l'usager demande le rôle) ; (2) DOIT livrer la notification de changement de l'état "unsubscribed" à toutes les ressources disponibles de l'usager ; et (3) DOIT livrer la présence indisponible à toutes les ressources disponibles de l'usager :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='contact@exemple.org'
       subscription='none'
       name='MyContact'>
      <group>MyBuddies</group>
    </item>
  </query>
</iq>

<presence
    from='contact@exemple.org'
    to='user@exemple.com'
    type='unsubscribed'/>

<presence
    from='contact@exemple.org/resource'
    to='user@exemple.com'
    type='unavailable'/>
```

4. À réception de la strophe de présence de type "unsubscribed", l'usager DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "unsubscribe" au contact, soit en la "refusant" par l'envoi d'une strophe de présence de type "subscribe" au contact ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur de l'usager qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement à l'usager (voir le paragraphe 9.4).

8.5.2 Cas n°2 : annulation lorsque l'abonnement est mutuel

Dans le second cas, l'usager a un abonnement aux informations de présence du contact et le contact a aussi un abonnement aux informations de présence de l'usager (c'est-à-dire, l'abonnement est mutuel).

1. Si le contact veut annuler l'abonnement de l'usager, il DOIT envoyer une strophe de présence de type "unsubscribed" à l'usager:

```
<presence to='usager@exemple.com' type='unsubscribed'/>
```

2. Par suite, le serveur du contact (1) DOIT envoyer une poussée de rôle à toutes les ressources disponibles du contact qui ont demandé le rôle, contenant un élément de rôle mis à jour pour l'usager avec l'attribut 'subscription' réglé à une valeur de "to", (2) DOIT acheminer la strophe de présence de type "unsubscribed" à l'usager, en marquant d'abord l'adresse 'from' comme JID nu (<contact@exemple.org>) du contact, et (3) DEVRAIT envoyer une présence indisponible à partir de toutes les ressources disponibles du contact à toutes les ressources disponibles de l'usager :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='user@exemple.com'
       subscription='to'
       name='SomeUser'>
      <group>SomeGroup</group>
    </item>
  </query>
</iq>

<presence
    from='contact@exemple.org'
    to='user@exemple.com'
    type='unsubscribed'/>
```

```
<presence
   from='contact@exemple.org/resource'
   to='user@exemple.com'
   type='unavailable'/>
```

3. À réception de la strophe de présence de type "unsubscribed" adressée à l'usager, le serveur de l'usager (1) DOIT initier une poussée de rôle à toutes les ressources disponibles de l'usager qui ont demandé le rôle, contenant un élément de rôle mis à jour pour le contact avec l'attribut 'subscription' réglé à une valeur de "from" (si l'usager est indisponible ou n'a pas demandé le rôle, le serveur de l'usager DOIT modifier l'élément de rôle et envoyer cet élément modifié la prochaine fois que l'usager demande le rôle) et (2) DOIT livrer la notification de changement d'état "unsubscribed" à toutes les ressources disponibles de l'usager, et (3) DOIT livrer la présence indisponible à toutes les ressources disponibles de l'usager :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='contact@exemple.org'
       subscription='from'
       name='MyContact'>
      <group>MyBuddies</group>
    </item>
  </query>
</iq>

<presence
   from='contact@exemple.org'
   to='user@exemple.com'
   type='unsubscribed'/>

<presence
   from='contact@exemple.org/resource'
   to='user@exemple.com'
   type='unavailable'/>
```

4. À réception de la strophe de présence de type "unsubscribed", l'usager DEVRAIT accuser réception de cette notification d'état d'abonnement soit en "l'affirmant" par l'envoi d'une strophe de présence de type "unsubscribe" au contact, soit en la "refusant" par l'envoi d'une strophe de présence de type "subscribe" au contact ; cette étape n'affecte pas nécessairement l'état d'abonnement (voir les détails à la Section 9 "États d'abonnement") mais fait plutôt savoir au serveur de l'usager qu'il NE DOIT PLUS envoyer de notification de changement de l'état d'abonnement à l'usager (voir le paragraphe 9.4).

Note : Évidemment il ne résulte pas de ceci une suppression de l'élément de rôle du rôle du contact, et le contact a toujours un abonnement aux informations de présence de l'usager. Afin d'annuler complètement un abonnement mutuel et supprimer pleinement l'élément de rôle du rôle du contact, le contact devrait mettre à jour l'élément de rôle avec un subscription='remove' comme défini au paragraphe 8.6 "Suppression d'un élément de rôle et annulation de tous les abonnements".

8.6 Suppression d'un élément de rôle et annulation de tous les abonnements

Comme il peut y avoir de nombreuses étapes impliquées dans la suppression complète d'un élément de rôle et l'annulation des abonnements dans les deux directions, le protocole de gestion de rôle comporte une méthode "raccourcie" pour le faire. Le processus qui peut être initié quel que soit l'état d'abonnement actuel est d'envoyer un ensemble de rôle contenant un élément pour le contact avec l'attribut 'subscription' réglé à une valeur de "remove" :

```
<iq type='set' id='remove1'>
  <query xmlns='jabber:iq:roster'>
    <item
       jid='contact@exemple.org'
       subscription='remove'/>
  </query>
</iq>
```

Lorsque l'usager supprime un contact de son rôle en réglant l'attribut 'subscription' à une valeur de "remove", le serveur de

l'usager (1) DOIT automatiquement annuler tout abonnement existant entre l'usager et le contact ('to' et 'from' à la fois comme approprié) ; (2) DOIT supprimer l'élément du rôle du rôle de l'usager et informer toutes les ressources disponibles de l'usager qui ont demandé la suppression de l'élément du rôle ; (3) DOIT informer la ressource qui a initié la suppression du succès, et (4) DEVRAIT envoyer au contact une présence indisponible à partir de toutes les ressources disponibles de l'usager :

```
<presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='unsubscribe'/>

<presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='unsubscribed'/>

<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
        jid='contact@exemple.org'
        subscription='remove'/>
  </query>
</iq>

<iq type='result' id='remove1'/>

<presence
    from='user@exemple.com/resource'
    to='contact@exemple.org'
    type='unavailable'/>
```

À réception de la strophe de présence de type "unsubscribe", le serveur du contact (1) DOIT initier une poussée de rôle à toutes les ressources disponibles associées au contact qui ont demandé le rôle, contenant un élément de rôle mis à jour pour l'usager avec l'attribut 'subscription' réglé à une valeur de "to" (si le contact est indisponible ou n'a pas demandé le rôle, le serveur du contact DOIT modifier l'élément de rôle et envoyer cet élément modifié la prochaine fois que le contact demande le rôle) et (2) DOIT aussi livrer la notification de changement de l'état "unsubscribe" à toutes les ressources disponibles du contact :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
        jid='user@exemple.com'
        subscription='to'
        name='SomeUser'>
      <group>SomeGroup</group>
    </item>
  </query>
</iq>

<presence
    from='user@exemple.com'
    to='contact@exemple.org'
    type='unsubscribe'/>
```

À réception de la strophe de présence de type "unsubscribed", le serveur du contact (1) DOIT initier une poussée de rôle à toutes les ressources disponibles associées au contact qui ont demandé le rôle, contenant un élément de rôle mis à jour pour l'usager avec l'attribut 'subscription' réglé à une valeur de "none" (si le contact est indisponible ou n'a pas demandé le rôle, le serveur du contact DOIT modifier l'élément de rôle et envoyer cet élément modifié la prochaine fois que le contact demande le rôle) et (2) DOIT aussi livrer la notification du changement de l'état "unsubscribe" à toutes les ressources disponibles du contact :

```
<iq type='set'>
  <query xmlns='jabber:iq:roster'>
    <item
        jid='user@exemple.com'
```

```
        subscription='none'
        name='SomeUser'>
       <group>SomeGroup</group>
     </item>
   </query>
 </iq>

 <presence
     from='user@exemple.com'
     to='contact@exemple.org'
     type='unsubscribed'/>
```

À réception de la strophe de présence de type "unavailable" adressée au contact, le serveur du contact DOIT livrer la présence indisponible à toutes les ressources disponibles de l'usager :

```
<presence
    from='user@exemple.com/resource'
    to='contact@exemple.org'
    type='unavailable'/>
```

Note : Lorsque l'usager supprime le contact du rôle de l'usager, l'état final du rôle du contact est que l'usager est encore dans le rôle du contact avec un état d'abonnement de "none" ; pour supprimer complètement l'élément de rôle pour l'usager, le contact doit aussi envoyer une demande de suppression de rôle.

9. États d'abonnement

Cette section donne des informations détaillées sur les états d'abonnement et le traitement par le serveur des strophes de présence relatives à l'abonnement (c'est-à-dire, les strophes de présence de type "subscribe", "subscribed", "unsubscribe", et "unsubscribed").

9.1 États définis

Il y a neuf états d'abonnement possibles,qui sont décrits ici dans la perspective de l'usager (pas du contact) :

1. "None" = contact et usager ne sont pas abonnés l'un à l'autre, et ni l'un ni l'autre n'a demandé un abonnement à l'autre.

2. "None + Pending Out" = contact et usager ne sont pas abonnés l'un à l'autre, et l'usager a envoyé au contact une demande d'abonnement mais le contact n'a pas encore répondu.

3. "None + Pending In" = contact et usager ne sont pas abonnés l'un à l'autre, et le contact a envoyé à l'usager une demande d'abonnement mais l'usager n'a pas encore répondu (noter que le serveur du contact NE DEVRAIT PAS pousser ou livrer des éléments de rôle dans cet état, mais DEVRAIT plutôt attendre que le contact ait approuvé la demande d'abonnement de l'usager)

4. "None + Pending Out/In" = contact et usager ne sont pas abonnés l'un à l'autre, le contact a envoyé à l'usager une demande d'abonnement mais l'usager n'a pas encore répondu, et l'usager a envoyé au contact une demande d'abonnement mais le contact n'a pas encore répondu.

5. "To" = l'usager est abonné au contact (une seule direction).

6. "To + Pending In" = l'usager est abonné au contact, et le contact a envoyé à l'usager une demande d'abonnement mais l'usager n'a pas encore répondu.

7. "From" = le contact est abonné à l'usager (une seule direction).

8. "From + Pending Out" = le contact est abonné à l'usager, et l'usager a envoyé au contact une demande d'abonnement mais le contact n'a pas encore répondu.

9. "Both" = l'usager et le contact sont abonnés l'un à l'autre (dans les deux sens).

9.2 Traitement par le serveur des strophes sortantes d'abonnement à Présence

Les strophes sortantes d'abonnement de présence permettent à l'usager de gérer son abonnement aux informations de présence du contact (via les types "subscribe" et "unsubscribe") et de gérer l'accès du contact aux informations de présence de l'usager (via les types "subscribed" et "unsubscribed").

Comme il est possible au serveur de l'usager et au serveur du contact de perdre la synchronisation concernant les états d'abonnement, le serveur de l'usager DOIT sans exception acheminer toutes les strophes de présence sortantes de type "subscribe" ou "unsubscribe" au contact afin que l'usager soit capable de resynchroniser son abonnement aux informations de présence du contact si nécessaire.

Le serveur de l'usager NE DEVRAIT PAS acheminer de strophe de présence de type "subscribed" ou "unsubscribed" au contact si la strophe ne résulte pas en un changement d'état d'abonnement du point de vue de l'usager, et NE DOIT PAS faire un changement d'état. Si la strophe résulte en un changement d'état d'abonnement, le serveur de l'usager DOIT acheminer la strophe au contact et DOIT faire le changement d'état approprié. Ces règles sont résumés dans les tableaux suivants.

Tableau 1 : Traitement recommandé des strophes "subscribed" sortantes

État existant	Acheminer ?	Nouvel état
"None"	non	pas de changement d'état
"None + Pending Out"	non	pas de changement d'état
"None + Pending In"	oui	"From"
"None + Pending Out/In"	oui	"From + Pending Out"
"To"	non	pas de changement d'état
"To + Pending In"	oui	"Both"
"From"	non	pas de changement d'état
"From + Pending Out"	non	pas de changement d'état
"Both"	non	pas de changement d'état

Tableau 2 : Traitement recommandé des strophes "unsubscribed" sortantes

État existant	Acheminer ?	Nouvel état
"None"	non	pas de changement d'état
"None + Pending Out"	non	pas de changement d'état
"None + Pending In"	oui	"None"
"None + Pending Out/In"	oui	"None + Pending Out"
"To"	non	pas de changement d'état
"To + Pending In"	oui	"To"
"From"	oui	"None"
"From + Pending Out"	oui	"None + Pending Out"
"Both"	oui	"To"

9.3 Traitement par le serveur des strophes entrantes d'abonnement à Présence

Les strophes d'abonnement de présence entrantes demandent ne action en rapport avec l'abonnement de la part de l'usager (via le type "subscribe") informent l'usager d'actions en rapport avec l'abonnement prises par le contact (via le type "unsubscribe") ou permettent au contact de gérer l'accès de l'usager aux informations de présence du contact (via les types "subscribed" et "unsubscribed").

Lorsque le serveur de l'usager reçoit une demande d'abonnement à l'usager de la part du contact (c'est-à-dire, une strophe de présence de type "subscribe") il DOIT livrer cette demande à l'usager pour approbation si l'usager n'a pas déjà accordé au contact l'accès aux informations de présence de l'usager et si il n'y a pas de demande d'abonnement entrante en cours ; cependant, le serveur de l'usager NE DEVRAIT PAS livrer la nouvelle demande si il y a une demande d'abonnement entrante en cours, car la précédente demande d'abonnement aura été enregistrée. Si l'usager a déjà accordé l'accès au contact pour les informations de présence de l'usager, le serveur de l'usager DEVRAIT auto répondre à une strophe de présence entrante de type "subscribe" provenant du contact par l'envoi d'une strophe de présence de type "subscribed" au contact au nom de l'usager ; cette règle permet au contact de resynchroniser l'état d'abonnement si nécessaire. Ces règles sont résumées dans le tableau suivant.

Tableau 3 : Traitement recommandé des strophes"subscribe" entrantes

État existant	Livrer ?	Nouvel état
"None"	oui	"None + Pending In"
"None + Pending Out"	oui	"None + Pending Out/In"
"None + Pending In"	non	pas de changement d'état
"None + Pending Out/In"	non	pas de changement d'état
"To"	oui	"To + Pending In"
"To + Pending In"	non	pas de changement d'état
"From"	non *	pas de changement d'état
"From + Pending Out"	non *	pas de changement d'état
"Both"	non *	pas de changement d'état

* Le serveur DEVRAIT auto répondre par une strophe "subscribed".

Lorsque le serveur de l'usager reçoit une strophe de présence de type "unsubscribe" pour l'usager de la part du contact, si la strophe résulte en un changement d'état d'abonnement du point de vue de l'usager, alors le serveur de l'usager DEVRAIT auto répondre en envoyant une strophe de présence de type "unsubscribed" au contact au nom de l'usager, DOIT livrer la strophe "unsubscribe" à l'usager, et DOIT changer d'état. Si aucun changement d'état d'abonnement n'en résulte, le serveur de l'usager NE DEVRAIT PAS livrer la strophe et NE DOIT PAS changer l'état. Ces règles sont résumées dans le tableau qui suit.

Tableau 4 : Traitement recommandé des strophes "unsubscribe" entrantes

État existant	Livrer ?	Nouvel état
"None"	non	pas de changement d'état
"None + Pending Out"	non	pas de changement d'état
"None + Pending In"	oui *	"None"
"None + Pending Out/In"	oui *	"None + Pending Out"
"To"	non	pas de changement d'état
"To + Pending In"	oui *	"To"
"From"	oui *	"None"
"From + Pending Out"	oui *	"None + Pending Out
"Both"	oui *	"To"

* Le serveur DEVRAIT auto répondre par une strophe "unsubscribed".

Lorsque le serveur de l'usager reçoit une strophe de présence de type "subscribed" pour l'usager de la part du contact, il NE DOIT PAS livrer la strophe à l'usager et NE DOIT PAS changer l'état d'abonnement si il n'y a pas une demande sortante en cours pour l'accès aux informations de présence du contact. Si il y a une demande sortante en cours pour l'accès aux informations de présence du contact et si la strophe de présence entrante de type "subscribed" résulte en un changement d'état d'abonnement, le serveur de l'usager DOIT livrer la strophe à l'usager et DOIT changer l'état d'abonnement. Si l'usager a déjà l'accès aux informations de présence du contact, la strophe de présence entrante de type "subscribed" ne résulte pas en un changement d'état d'abonnement ; donc, le serveur de l'usager NE DEVRAIT PAS livrer la strophe à l'usager et NE DOIT PAS changer l'état d'abonnement. Ces règles sont résumées dans le tableau suivant.

Tableau 5 : Traitement recommandé des strophes "subscribed" entrantes

État existant	Livrer ?	Nouvel état
"None"	non	pas de changement d'état
"None + Pending Out"	oui	"To"
"None + Pending In"	non	pas de changement d'état
"None + Pending Out/In"	oui	"To + Pending In"
"To"	non	pas de changement d'état
"To + Pending In"	non	pas de changement d'état
"From"	non	pas de changement d'état
"From + Pending Out"	oui	"Both"
"Both"	non	pas de changement d'état

Lorsque le serveur de l'usager reçoit une strophe de présence de type "unsubscribed" pour l'usager de la part du contact, il DOIT livrer la strophe à l'usager et DOIT changer l'état d'abonnement si il y a une demande sortante en cours pour accéder aux informations de présence du contact ou si l'usager a actuellement accès aux informations de présence du contact. Autrement, le serveur de l'usager NE DEVRAIT PAS livrer la strophe et NE DOIT PAS changer l'état d'abonnement. Ces règles sont résumées dans le tableau suivant.

Tableau 6 : Traitement recommandé des strophes "unsubscribed" entrantes

État existant	Livrer ?	Nouvel état
"None"	non	pas de changement d'état
"None + Pending Out"	oui	"None"
"None + Pending In"	non	pas de changement d'état
"None + Pending Out/In"	oui	"None + Pending In"
"To"	oui	"None"
"To + Pending In"	oui	"None + Pending In"
"From"	non	pas de changement d'état
"From + Pending Out"	oui	"From"
"Both"	oui	"From"

9.4 Livraison par le serveur et acquittement par le client des demandes d'abonnement et des notifications de changement d'état

Lorsque un serveur reçoit une strophe de présence entrante de type "subscribe" (c'est-à-dire, une demande d'abonnement) ou de type "subscribed", "unsubscribe", ou "unsubscribed" (c'est-à-dire, une notification de changement d'état d'abonnement) en plus d'envoyer la poussée de rôle appropriée (ou de mise à jour de rôle lorsque le rôle est ensuite demandée par une ressource disponible) il DOIT livrer la demande ou notification au receveur désigné au moins une fois. Un serveur PEUT exiger que le receveur accuse réception de toutes les notifications de changement d'état (et DOIT exiger l'accusé de réception dans le cas de demandes d'abonnement, c'est-à-dire, des strophes de présence de type "subscribe"). Pour exiger l'accusé de réception, un serveur DEVRAIT envoyer la demande ou notification au receveur chaque fois que le receveur se connecte, jusqu'à ce que le receveur accuse réception de la notification en "affirmant" ou "refusant" la notification, comme le montre le tableau suivant:

Tableau 7 : Accusé de réception des notifications de changement d'état d'abonnement

Type de strophe	Accepte	Refuse
subscribe	subscribed	unsubscribed
subscribed	subscribe	unsubscribe
unsubscribe	unsubscribed	subscribed
unsubscribed	unsubscribe	subscribe

Évidemment, étant donnés les tableaux d'état d'abonnement ci-dessus, certaines des strophes d'accusé de réception seront acheminées au contact et résulteront en des changements de l'état d'abonnement, tandis que d'autres ne le feront pas. Cependant, toutes ces strophes DOIVENT résulter en ce que le serveur n'envoie plus de notification d'état d'abonnement à l'usager.

Parce qu'un serveur d'usager DOIT automatiquement générer des strophes de présence sortantes de type "unsubscribe" et "unsubscribed" à réception d'un ensemble de rôle avec l'attribut 'subscription' réglé à une valeur de "remove" (voir au paragraphe 8.6 "Suppression d'un élément de rôle et annulation de tous les abonnements") le serveur DOIT traiter une demande de suppression de rôle comme équivalente à l'envoi de deux de ces strophes de présence afin de déterminer si il continue d'envoyer les notifications de changement d'état d'abonnement de type "subscribe" ou "subscribed" à l'usager.

10. Blocage de communication

La plupart des systèmes de messagerie instantanée ont trouvé qu'il était nécessaire de mettre en œuvre une méthode pour que les utilisateurs bloquent des communications provenant d'autres usagers particuliers (ceci est aussi exigé aux paragraphes 5.1.5, 5.1.15, 5.3.2, et 5.4.10 de la [RFC2779]). Dans XMPP ceci se fait par la gestion de listes de confidentialité en utilisant l'espace de noms 'jabber:iq:privacy'.

Les listes de confidentialité côté serveur permettent de mener à bien les cas d'utilisation suivants :
- o Restituer les listes de confidentialité de quelqu'un.
- o Ajouter, supprimer, et éditer les listes de confidentialité de quelqu'un.
- o Établir, changer, ou refuser des listes actives.
- o Établir, changer, ou refuser la liste par défaut (c'est-à-dire, la liste qui est active par défaut).
- o Permettre ou bloquer les messages sur la base du JID, du groupe, ou du type d'abonnement (ou globalement).
- o Permettre ou bloquer les notifications de présence entrantes sur la base du JID, groupe, ou type d'abonnement (ou globalement).

- o Permettre ou bloquer les notifications de présence sortantes sur la base du JID, groupe, ou type d'abonnement (ou globalement).
- o Permettre ou bloquer les strophes IQ sur la base du JID, groupe, ou type d'abonnement (ou globalement).
- o Permettre ou bloquer toutes les communications sur la base du JID, groupe, ou type d'abonnement (ou globalement).

Note : les notifications de présence n'incluent pas d'abonnements de présence, seulement les informations de présence qui sont diffusées aux entités qui sont abonnées aux informations de présence d'un usager. Donc cela inclut seulement les strophes de présence sans attribut 'type' ou de type='unavailable'.

10.1 Syntaxe et sémantique

Un usager PEUT définir une ou plusieurs listes de confidentialité qui sont mémorisées par le serveur de l'usager. Chaque élément <list/> contient une ou plusieurs règles sous la forme d'éléments <item/>, et chaque élément <item/> utilise des attributs pour définir un type de règle de confidentialité, une valeur spécifique à laquelle la règle s'applique, les actions pertinentes, et la place de l'élément dans l'ordre de traitement.

La syntaxe est la suivante :

```
<iq>
  <query xmlns='jabber:iq:privacy'>
    <list name='foo'>
      <item
        type='[jid|group|subscription]'
        value='bar'
        action='[allow|deny]'
        order='unsignedInt'>
       [<message/>]
       [<presence-in/>]
       [<presence-out/>]
       [<iq/>]
      </item>
    </list>
  </query>
</iq>
```

Si le type est "jid", alors l'attribut 'value' DOIT contenir un identifiant Jabber valide. Les JID DEVRAIENT être confrontés dans l'ordre suivant :

1. <usager@domaine/ressource> (seule cette ressource correspond)
2. <usager@domaine> (toute ressource correspond)
3. <domaine/ressource> (seule cette ressource correspond)
4. <domaine> (le domaine lui-même correspond, comme le fait tout usager@domaine, domaine/ressource, ou adresse contenant un sous domaine).

Si le type est "group", alors l'attribut 'value' DEVRAIT contenir le nom d'un groupe dans le rôle de l'usager. (Si un client tente de mettre à jour, créer, ou supprimer un élément de liste avec un groupe qui n'est pas dans le rôle de l'usager, le serveur DEVRAIT retourner au client une erreur de strophe de <item-not-found/>.)

Si le type est "subscription", alors l'attribut 'value' DOIT être un de "both", "to", "from", ou "none" comme défini au paragraphe 7.1 "Syntaxe et sémantique de rôle", où "none" inclut des entités qui sont totalement inconnues de l'usager et donc pas du tout sur le rôle de l'usager.

Si aucun attribut 'type' n'est inclus, la règle donne le cas "fall-through" où on n'arrive à rien.

L'attribut 'action' DOIT être inclus et sa valeur DOIT être "allow" ou "deny".

L'attribut 'order' DOIT être inclus et sa valeur DOIT être un entier non négatif qui soit unique parmi tous les éléments de la liste. (Si un client tente de créer ou mettre à jour une liste avec des valeurs d'ordre non uniques, le serveur DOIT retourner au client une erreur de strophe <bad-request/>.)

L'élément <item/> PEUT contenir un ou plusieurs éléments fils qui permettent à une entité de spécifier un contrôle d'une granularité plus fine sur la sorte de strophes à bloquer (c'est-à-dire, plutôt que de bloquer toutes les strophes). Les éléments fils admissibles sont :

- o <message/> -- bloque les strophes de message entrantes
- o <iq/> -- bloque les strophes IQ entrantes
- o <presence-in/> -- bloque les notifications de présence entrantes
- o <presence-out/> -- bloque les notifications de présence sortantes

Au sein de l'espace de noms 'jabber:iq:privacy', l'enfant <query/> d'une strophe IQ de type "set" NE DOIT PAS inclure plus d'un élément fils (c'est-à-dire, la strophe DOIT contenir seulement un élément <active/>, un élément <default/>, ou un élément <list/>) ; si une entité envoyeuse viole cette règle, l' entité receveuse DOIT retourner une erreur de strophe <bad-request/>.

Lorsque un client ajoute ou met à jour une liste de confidentialité, l'élément <list/> DEVRAIT contenir au moins un élément fils <item/> ; lorsque un client supprime une liste de confidentialité, l'élément <list/> NE DOIT PAS contenir d'élément fils <item/>.

Lorsque un client met à jour une liste de confidentialité, il doit inclure tous les éléments désirés (c'est-à-dire, pas un "delta").

10.2. Règles de traitement

1. Si une liste active est établie pour une session, cela affecte seulement la ou les sessions pour lesquelles elle est activée, et seulement pour la durée de la ou des sessions ; le serveur DOIT appliquer seulement la liste active et NE DOIT PAS appliquer la liste par défaut (c'est-à-dire, il n'y a pas de "mise en couche" de listes).

2. La liste par défaut s'applique à l'usager comme un tout, et est traitée si il n'y a pas de liste active établie pour la session/ressource cible à laquelle une strophe est adressée, ou si il n'y a pas de session en cours pour l'usager.

3. Si aucune liste active n'est établie pour une session (ou si il n'y a pas de session en cours pour l'usager) et si il n'y a pas de liste par défaut, alors toutes les strophes DEVRAIENT être acceptées ou traitées de façon appropriée par le serveur au nom de l'usager conformément aux "Règles de traitement des strophes XML par le serveur" (Section 11).

4. Les listes de confidentialité DOIVENT être la première règle de livraison appliquée par un serveur, supplantant (1) les règles d'acheminement et de livraison spécifiées dans "Règles de traitement des strophes XML par le serveur" (Section 11), et (2) le traitement de strophe de présence en relation avec l'abonnement (et la génération de poussées de rôle correspondantes) spécifié dans "Intégration d'éléments de rôle et abonnements à présence" (Section 8).

5. L'ordre dans lequel les éléments de liste de confidentialité sont traités par le serveur est important. Les éléments de liste DOIVENT être traités en ordre ascendant déterminé par les valeurs d'entier de l'attribut 'order' pour chaque <item/>.

6. Aussitôt qu'une strophe est confrontée à une règle de liste de confidentialité, le serveur DOIT traiter la strophe de la façon appropriée conformément à la règle et cesser le traitement.

7. Si aucun élément d'abandon n'est fourni dans une liste, l'action d'abandon est supposée être "allow" *(permettre)*.

8. Si un usager met à jour la définition pour une liste active, le traitement suivant fondé sur cette liste active DOIT utiliser la définition mise à jour (pour toutes les ressources auxquelles cette liste active s'applique actuellement).

9. Si un changement à l'état d'abonnement ou au groupe de rôle d'un élément de rôle défini dans une liste active ou par défaut survient durant une session d'un usager, le traitement suivant fondé sur cette liste DOIT prendre en compte le changement d'état ou de groupe (pour toutes les ressources auxquelles cette liste s'applique actuellement).

10. Lorsque la définition d'une règle est modifiée, le serveur DOIT envoyer une strophe IQ de type "set" à toutes les ressources connectées, contenant un élément <query/> avec seulement un élément fils <list/>, où l'attribut 'name' est réglé au nom de la liste de confidentialité modifiée. Ces " poussées de liste de confidentialité" adhèrent à la même sémantique que les "poussées de rôle" utilisées dans la gestion de rôle, excepté que seul le nom de liste (et non la définition complète de la liste ou le "delta") est poussé aux ressources connectées. Il appartient à la ressource receveuse de déterminer si il faut restituer la définition de liste modifiée, mais une ressource connectée DEVRAIT le faire si la liste s'applique actuellement à elle.

11. Lorsque une ressource tente de supprimer une liste ou de spécifier une nouvelle liste par défaut alors que cette liste s'applique à une ressource connectée autre que la ressource envoyeuse, le serveur DOIT retourner une erreur <conflict/> à la ressource envoyeuse et NE DOIT PAS faire le changement demandé.

10.3 Restitution des listes de confidentialité de quelqu'un

Exemple : Le client demande les noms de liste de confidentialités au serveur :

```
<iq from='romeo@exemple.net/orchard' type='get' id='getlist1'>
  <query xmlns='jabber:iq:privacy'/>
</iq>
```

Exemple : Le serveur envoie les noms de la liste de confidentialité au client, précédés de la liste active et de la liste par défaut :

```
<iq type='result' id='getlist1' to='romeo@exemple.net/orchard'>
  <query xmlns='jabber:iq:privacy'>
    <active name='private'/>
    <default name='public'/>
    <list name='public'/>
    <list name='private'/>
    <list name='special'/>
  </query>
</iq>
```

Exemple : Le client demande une liste de confidentialité au serveur :

```
<iq from='romeo@exemple.net/orchard' type='get' id='getlist2'>
  <query xmlns='jabber:iq:privacy'>
    <list name='public'/>
  </query>
</iq>
```

Exemple : Le serveur envoie une liste de confidentialité au client :

```
<iq type='result' id='getlist2' to='romeo@exemple.net/orchard'>
  <query xmlns='jabber:iq:privacy'>
    <list name='public'>
      <item type='jid'
            value='tybalt@exemple.com'
            action='deny'
            order='1'/>
      <item action='allow' order='2'/>
    </list>
  </query>
</iq>
```

Exemple : Le client demande une autre liste de confidentialité au serveur :

```
<iq from='romeo@exemple.net/orchard' type='get' id='getlist3'>
  <query xmlns='jabber:iq:privacy'>
    <list name='private'/>
  </query>
</iq>
```

Exemple : Le serveur envoie une autre liste de confidentialité au client :

```
<iq type='result' id='getlist3' to='romeo@exemple.net/orchard'>
  <query xmlns='jabber:iq:privacy'>
    <list name='private'>
      <item type='subscription'
            value='both'
            action='allow'
            order='10'/>
       <item action='deny' order='15'/>
    </list>
  </query>
</iq>
```

Exemple : Le client demande encore une autre liste de confidentialité au serveur :

```
<iq from='romeo@exemple.net/orchard' type='get' id='getlist4'>
  <query xmlns='jabber:iq:privacy'>
    <list name='special'/>
  </query>
</iq>
```

Exemple : Le serveur envoie encore une autre liste de confidentialité au client :

```
<iq type='result' id='getlist4' to='romeo@exemple.net/orchard'>
  <query xmlns='jabber:iq:privacy'>
    <list name='special'>
      <item type='jid'
            value='juliet@exemple.com'
            action='allow'
            order='6'/>
      <item type='jid'
            value='benvolio@exemple.org'
            action='allow'
            order='7'/>
      <item type='jid'
            value='mercutio@exemple.org'
            action='allow'
            order='42'/>
      <item action='deny' order='666'/>
    </list>
  </query>
</iq>
```

Dans cet exemple, l'usager a trois listes : (1) 'public', qui permet les communications de tout le monde sauf une entité spécifique (c'est la liste par défaut) ; (2) 'private', qui permet les communications seulement avec les contacts qui ont un abonnement bidirectionnel avec l'usager (c'est la liste active) ; et (3) 'special', qui permet les communications seulement avec trois entités spécifiques.

Si l'usager tente de restituer une liste mais que la liste de ce nom n'existe pas, le serveur DOIT retourner une erreur de strophe <item-not-found/> à l'usager :

Exemple : Le client tente de restituer une liste non existante :

```
<iq to='romeo@exemple.net/orchard' type='error' id='getlist5'>
  <query xmlns='jabber:iq:privacy'>
    <list name='The Empty Set'/>
  </query>
  <error type='cancel'>
    <item-not-found
      xmlns='urn:ietf:params:xml:ns:xmpp-strophes'/>
  </error>
</iq>
```

Il n'est permis à l'usager de restituer qu'une seule liste à la fois. Si l'usager tente de récupérer plus d'une liste dans la même demande, le serveur DOIT retourner une erreur de strophe <bad request/> à l'usager :

Exemple : Le client tente de récupérer plus d'une liste :

```
<iq to='romeo@exemple.net/orchard' type='error' id='getlist6'>
  <query xmlns='jabber:iq:privacy'>
    <list name='public'/>
    <list name='private'/>
    <list name='special'/>
  </query>
  <error type='modify'>
```

```
    <bad-request
        xmlns='urn:ietf:params:xml:ns:xmpp-strophes'/>
  </error>
</iq>
```

10.4 Gestion des listes actives

Pour établir ou changer la liste active actuellement appliquée par le serveur, l'usager DOIT envoyer une strophe IQ de type "set" avec un élément <query/> qualifié par l'espace de noms 'jabber:iq:privacy' qui contient un élément fils <active/> vide possédant un attribut 'name' dont la valeur est réglée au nom de liste désiré.

Exemple : Le client demande à changer de liste active :

```
<iq from='romeo@exemple.net/orchard' type='set' id='active1'>
  <query xmlns='jabber:iq:privacy'>
    <active name='special'/>
  </query>
</iq>
```

Le serveur DOIT activer et appliquer la liste demandée avant de renvoyer le résultat au client.

Exemple : Le serveur accuse réception du succès de changement de liste active :

```
<iq type='result' id='active1' to='romeo@exemple.net/orchard'/>
```

Si l'usager tente d'établir une liste active mais si cette liste a un nom qui n'existe pas, le serveur DOIT retourner une erreur de strophe <item-not-found/> à l'usager.

Exemple : Le client tente d'établir comme active une liste qui n'existe pas :

```
<iq to='romeo@exemple.net/orchard' type='error' id='active2'>
  <query xmlns='jabber:iq:privacy'>
    <active name='The Empty Set'/>
  </query>
  <error type='cancel'>
    <item-not-found
        xmlns='urn:ietf:params:xml:ns:xmpp-strophes'/>
  </error>
</iq>
```

Pour refuser l'utilisation d'une liste active, la ressource connectée DOIT envoyer un élément <active/> vide sans attribut 'name'.

Exemple : Le client refuse l'utilisation des listes actives :

```
<iq from='romeo@exemple.net/orchard' type='set' id='active3'>
  <query xmlns='jabber:iq:privacy'>
    <active/>
  </query>
</iq>
```

Exemple : Le serveur accuse réception du succès du refus de toute liste active :

```
<iq type='result' id='active3' to='romeo@exemple.net/orchard'/>
```

10.5 Gestion de la liste par défaut

Pour changer sa liste par défaut (qui s'applique à l'usager comme un tout, et pas seulement à la ressource envoyeuse) l'usager DOIT envoyer une strophe IQ de type "set" avec un élément <query/> qualifié par l'espace de noms 'jabber:iq:privacy' qui contient un élément fils <default/> vide possédant un attribut 'name' dont la valeur est réglée au nom de liste désiré.

Exemple : L'usager demande à changer de liste par défaut :

```
<iq from='romeo@exemple.net/orchard' type='set' id='default1'>
  <query xmlns='jabber:iq:privacy'>
    <default name='special'/>
  </query>
</iq>
```

Exemple : Le serveur accuse réception du succès du changement de liste par défaut :

```
<iq type='result' id='default1' to='romeo@exemple.net/orchard'/>
```

Si l'usager tente de changer la liste qui est la liste par défaut mais si la liste par défaut est utilisée par au moins une ressource connectée autre que la ressource envoyeuse, le serveur DOIT retourner une erreur de strophe <conflict/> à la ressource envoyeuse.

Exemple : Le client tente de changer la liste par défaut mais cette liste est utilisée par une autre ressource :

```
<iq to='romeo@exemple.net/orchard' type='error' id='default1'>
  <query xmlns='jabber:iq:privacy'>
    <default name='special'/>
  </query>
  <error type='cancel'>
    <conflict     xmlns='urn:ietf:params:xml:ns:xmpp-strophes'/>
  </error>
</iq>
```

Si l'usager tente d'établir une liste par défaut mais si une liste de ce nom n'existe pas, le serveur DOIT retourner une erreur de strophe <item-not-found/> à l'usager.

Exemple : Le client tente d'établir comme liste par défaut une liste qui n'existe pas :

```
<iq to='romeo@exemple.net/orchard' type='error' id='default1'>
  <query xmlns='jabber:iq:privacy'>
    <default name='The Empty Set'/>
  </query>
  <error type='cancel'>
    <item-not-found
       xmlns='urn:ietf:params:xml:ns:xmpp-strophes'/>
  </error>
</iq>
```

Afin de refuser l'utilisation d'une liste par défaut (c'est-à-dire, d'utiliser tout le temps les règles d'acheminement de strophe du domaine) l'usager DOIT envoyer l'élément <default/> vide sans attribut 'name'.

Exemple : Le client refuse d'utiliser la liste par défaut :

```
<iq from='romeo@exemple.net/orchard' type='set' id='default2'>
  <query xmlns='jabber:iq:privacy'>
    <default/>
  </query>
</iq>
```

Exemple : Le serveur accuse réception du succès du refus de toute liste par défaut :

```
<iq type='result' id='default2' to='romeo@exemple.net/orchard'/>
```

Si une ressource connectée tente de refuser l'utilisation d'une liste par défaut pour l'usager comme un tout mais si la liste par défaut s'applique actuellement au moins à une autre ressource connectée, le serveur DOIT retourner une erreur <conflict/> à la ressource envoyeuse.

Exemple : Le client tente de refuser une liste par défaut mais cette liste est utilisée par une autre ressource :

```
<iq to='romeo@exemple.net/orchard' type='error' id='default3'>
```

```
  <query xmlns='jabber:iq:privacy'>
    <default/>
  </query>
  <error type='cancel'>
    <conflict
        xmlns='urn:ietf:params:xml:ns:xmpp-strophes'/>
  </error>
</iq>
```

10.6 Édition d'une liste de confidentialité

Pour éditer une liste de confidentialité, l'usager DOIT envoyer une strophe IQ de type "set" avec un élément <query/> qualifié par l'espace de noms 'jabber:iq:privacy' qui contient un élément fils <list/> possédant un attribut 'name' dont la valeur est réglée au nom de liste que l'usager aimerait éditer. L'élément <list/> DOIT contenir un ou plusieurs éléments <item/> qui spécifient les changements désirés par l'usager à la liste en incluant tous les éléments dans la liste (pas le "delta").

Exemple : Le client édite une liste de confidentialité :

```
<iq from='romeo@exemple.net/orchard' type='set' id='edit1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='public'>
      <item type='jid'
            value='tybalt@exemple.com'
            action='deny'
            order='3'/>
      <item type='jid'
            value='paris@exemple.org'
            action='deny'
            order='5'/>
      <item action='allow' order='68'/>
    </list>
  </query>
</iq>
```

Exemple : Le serveur accuse réception du succès de l'édition de la liste :

```
<iq type='result' id='edit1' to='romeo@exemple.net/orchard'/>
```

Note : La valeur de l'attribut 'order' pour tout élément donné n'est pas fixée. Donc dans l'exemple précédent, si l'usager veut ajouter des éléments entre l'élément "tybalt@exemple.com" et l'élément "paris@exemple.org", le client de l'usager DOIT renuméroter les éléments pertinents avant de soumettre la liste au serveur.

Le serveur DOIT maintenant envoyer une "poussée de liste de confidentialité" à toutes les ressources connectées.

Exemple : Poussée d'une liste de confidentialité sur une édition de liste :

```
<iq to='romeo@exemple.net/orchard' type='set' id='push1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='public'/>
  </query>
</iq>

<iq to='romeo@exemple.net/home' type='set' id='push2'>
  <query xmlns='jabber:iq:privacy'>
    <list name='public'/>
  </query>
</iq>
```

Conformément à la sémantique des strophes IQ définie dans la [RFC3920], chaque ressource connectée DOIT aussi retourner un résultat IQ au serveur.

Exemple : Accusé de réception de poussée de liste de confidentialité :

```
<iq from='romeo@exemple.net/orchard'
   type='result'
   id='push1'/>

<iq from='romeo@exemple.net/home'
   type='result'
   id='push2'/>
```

10.7 Ajout d'une nouvelle liste de confidentialité

Pour créer une nouvelle liste, on utilise le même protocole que pour éditer une liste existante. Si le nom de la liste correspond à celui d'une liste existante, la demande d'ajout d'une nouvelle liste va écraser l'ancienne. Comme avec l'édition de liste, le serveur DOIT aussi envoyer une "poussée de liste de confidentialité" à toutes les ressources connectées.

10.8 Suppression d'une liste de confidentialité

Pour supprimer une liste de confidentialité, l'usager DOIT envoyer une strophe IQ de type "set" avec un élément <query/> qualifié par l'espace de noms 'jabber:iq:privacy' qui contient un élément fils <list/> vide possédant un attribut 'name' dont la valeur est réglée au nom de liste que l'usager aimerait supprimer.

Exemple : Le client supprime une liste de confidentialité :

```
<iq from='romeo@exemple.net/orchard' type='set' id='remove1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='private'/>
  </query>
</iq>
```

Exemple : Le serveur accuse réception du succès de la suppression de liste :

```
<iq type='result' id='remove1' to='romeo@exemple.net/orchard'/>
```

Si un usager tente de supprimer une liste qui est actuellement appliquée à au moins une ressource autre que la ressource envoyeuse, le serveur DOIT retourner une erreur de strophe <conflict/> à l'usager ; c'est-à-dire, l'usager DOIT d'abord établir une autre liste à active ou défaut avant de tenter de la supprimer. Si l'usager tente de supprimer une liste mais qu'une liste de ce nom n'existe pas, le serveur DOIT retourner une erreur de strophe <item-not-found/> à l'usager. Si l'usager tente de supprimer plus d'une liste dans la même demande, le serveur DOIT retourner une erreur de strophe <bad request/> à l'usager.

10.9 Blocage de messages

Les listes de confidentialité côté serveur permettent à un usager de bloquer les messages entrants provenant d'autres entités sur la base du JID de l'entité, du groupe de rôle, ou de l'état d'abonnement (ou globalement). Les exemples suivants illustrent le protocole. (Noter que pour faire court, les strophes IQ de type "result" ne sont pas montrées dans les exemples, ni les "poussées de liste de confidentialité".)

Exemple : L'usager bloque sur la base du JID :

```
<iq from='romeo@exemple.net/orchard' type='set' id='msg1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='message-jid-exemple'>
      <item type='jid'
            value='tybalt@exemple.com'
            action='deny'
            order='3'>
        <message/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra pas de messages de l'entité qui a le JID spécifié.

Exemple : L'usager bloque sur la base du groupe de rôle :

```
<iq from='romeo@exemple.net/orchard' type='set' id='msg2'>
 <query xmlns='jabber:iq:privacy'>
  <list name='message-group-exemple'>
   <item type='group'
         value='Enemies'
         action='deny'
         order='4'>
     <message/>
    </item>
  </list>
 </query></iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra de message d'aucune entité dans le groupe de rôle spécifié.

Exemple : L'usager bloque sur la base du type d'abonnement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='msg3'>
 <query xmlns='jabber:iq:privacy'>
  <list name='message-sub-exemple'>
   <item type='subscription'
         value='none'
         action='deny'
         order='5'>
     <message/>     </item>
  </list>
 </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra de message d'aucune entité qui a le type d'abonnement spécifié.

Exemple : L'usager bloque globalement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='msg4'>
 <query xmlns='jabber:iq:privacy'>
  <list name='message-global-exemple'>
   <item action='deny' order='6'>
    <message/>
   </item>
  </list>
 </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra de message d'aucun autre usager.

10.10 Blocage des notifications de présence entrantes

Les listes de confidentialité côté serveur permettent à un usager de bloquer les notifications de présence entrantes provenant d'autres entités sur la base du JID de l'entité, du groupe du rôle, ou de l'état d'abonnement (ou globalement). Les exemples suivants illustrent le protocole.

Note : Les notifications de présence n'incluent pas les abonnements de présence, seulement les informations de présence qui sont diffusées à l'usager parce que l'usager est actuellement abonné aux informations de présence d'un contact. Donc cela inclut seulement les strophe de présences sans attribut 'type' ou de type='unavailable'.

Exemple : L'usager bloque sur la base du JID :

```
<iq from='romeo@exemple.net/orchard' type='set' id='presin1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='presin-jid-exemple'>
      <item type='jid'
            value='tybalt@exemple.com'
            action='deny'
            order='7'>
        <presence-in/>
      </item>    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra pas de notifications de présence de l'entité qui a le JID spécifié.

Exemple : L'usager bloque sur la base du groupe du rôle :

```
<iq from='romeo@exemple.net/orchard' type='set' id='presin2'>
  <query xmlns='jabber:iq:privacy'>
    <list name='presin-group-exemple'>
      <item type='group'
            value='Enemies'
            action='deny'
            order='8'>
        <presence-in/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra pas de notifications de présence des entités du groupe de rôle spécifié.

Exemple : L'usager bloque sur la base du type d'abonnement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='presin3'>
  <query xmlns='jabber:iq:privacy'>
    <list name='presin-sub-exemple'>
      <item type='subscription'
            value='to'
            action='deny'
            order='9'>
        <presence-in/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra pas de notification de présence d'entités qui ont le type d'abonnement spécifié.

Exemple : L'usager bloque globalement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='presin4'>
  <query xmlns='jabber:iq:privacy'>
    <list name='presin-global-exemple'>
      <item action='deny' order='11'>
        <presence-in/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra pas de notifications de présence des autres usagers.

10.11 Blocage des notifications de Présence sortantes

Les listes de confidentialités côté serveur permettent à un usager de bloquer les notifications de présence sortantes aux autres entités sur la base du JID de l'entité, du groupe du rôle, ou de l'état d'abonnement (ou globalement). Les exemples suivants illustrent le protocole.

Note : Les notifications de présence n'incluent pas les abonnements à présence, seulement les informations de présence qui sont diffusées aux contacts parce que ces contacts sont actuellement abonnés aux informations de présence de l'usager. Donc cela inclut seulement des strophes de présence sans attribut 'type' ou de type='unavailable'.

Exemple : L'usager bloque sur la base de JID :

```
<iq from='romeo@exemple.net/orchard' type='set' id='presout1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='presout-jid-exemple'>
      <item type='jid'
            value='tybalt@exemple.com'
            action='deny'
            order='13'>
        <presence-out/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager n'enverra pas de notifications de présence à l'entité qui a le JID spécifié.

Exemple : L'usager bloque sur la base du groupe du rôle :

```
<iq from='romeo@exemple.net/orchard' type='set' id='presout2'>
  <query xmlns='jabber:iq:privacy'>
    <list name='presout-group-exemple'>
      <item type='group'
            value='Enemies'
            action='deny'
            order='15'>
        <presence-out/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager n'enverra de notifications de présence à aucune entité dans le groupe de rôle spécifié.

Exemple : L'usager bloque sur la base de type d'abonnement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='presout3'>
  <query xmlns='jabber:iq:privacy'>
    <list name='presout-sub-exemple'>
      <item type='subscription'
            value='from'
            action='deny'
            order='17'>
        <presence-out/>
      </item>
    </list>
```

```
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager n'enverra de notification de présence à aucune entité qui a le type d'abonnement spécifié.

Exemple : L'usager bloque globalement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='presout4'>
  <query xmlns='jabber:iq:privacy'>
    <list name='presout-global-exemple'>
      <item action='deny' order='23'>
        <presence-out/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager n'enverra de notification de présence à aucun des autres usagers.

10.12 Blocage des strophes IQ

Les listes de confidentialité côté serveur permettent à un usager de bloquer les strophes IQ entrantes provenant d'autres entités sur la base du JID de l'entité, du groupe du rôle, ou de l'état d'abonnement (ou globalement). Les exemples suivants illustrent le protocole.

Exemple : L'usager bloque sur la base du JID :

```
<iq from='romeo@exemple.net/orchard' type='set' id='iq1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='iq-jid-exemple'>
      <item type='jid'
            value='tybalt@exemple.com'
            action='deny'
            order='29'>
        <iq/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne va pas recevoir les strophes IQ de l'entité qui a le JID spécifié.

Exemple : L'usager bloque sur la base du groupe du rôle :

```
<iq from='romeo@exemple.net/orchard' type='set' id='iq2'>
  <query xmlns='jabber:iq:privacy'>
    <list name='iq-group-exemple'>
      <item type='group'
            value='Enemies'
            action='deny'
            order='31'>
        <iq/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne va recevoir les strophes IQ d'aucune entité qui est dans le groupe de rôle spécifié.

Exemple : L'usager bloque sur la base de type d'abonnement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='iq3'>
  <query xmlns='jabber:iq:privacy'>
    <list name='iq-sub-exemple'>
      <item type='subscription'
            value='none'
            action='deny'
            order='17'>
        <iq/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne va recevoir les strophes IQ d'aucune entité du type d'abonnement spécifié.
Exemple : L'usager bloque globalement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='iq4'>
  <query xmlns='jabber:iq:privacy'>
    <list name='iq-global-exemple'>
      <item action='deny' order='1'>
        <iq/>
      </item>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne recevra les strophes IQ d'aucun autre usager.

10.13 Blocage de toutes les communications

Les listes de confidentialité côté serveur permettent à un usager de bloquer toutes les strophes provenant des, ou destinées aux autres entités sur la base du JID de l'entité, du groupe du rôle, de l'état d'abonnement (ou globalement). Noter que ceci inclut les strophes de présence en rapport avec l'abonnement, qui sont exclues par le paragraphe 10.10 "Blocage des notifications de présence entrantes". Les exemples suivants illustrent le protocole.

Exemple : L'usager bloque sur la base du JID :

```
<iq from='romeo@exemple.net/orchard' type='set' id='all1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='all-jid-exemple'>
      <item type='jid'
            value='tybalt@exemple.com'
            action='deny'
            order='23'/>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne va recevoir aucune communication de, ni envoyer aucune strophe à l'entité qui a le JID spécifié.

Exemple : L'usager bloque sur la base du groupe du rôle :

```
<iq from='romeo@exemple.net/orchard' type='set' id='all2'>
  <query xmlns='jabber:iq:privacy'>
    <list name='all-group-exemple'>
      <item type='group'
            value='Enemies'
```

```
        action='deny'
        order='13'/>
   </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne va recevoir aucune communication de, ni envoyer aucune strophe à toute entité qui est dans le groupe de rôle spécifié.

Exemple : L'usager bloque sur la base de type d'abonnement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='all3'>
  <query xmlns='jabber:iq:privacy'>
    <list name='all-sub-exemple'>
      <item type='subscription'
          value='none'
          action='deny'
          order='11'/>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne va recevoir aucune communication de, ni envoyer aucune strophe à toute entité qui a le type d'abonnement spécifié.

Exemple : L'usager bloque globalement :

```
<iq from='romeo@exemple.net/orchard' type='set' id='all4'>
  <query xmlns='jabber:iq:privacy'>
    <list name='all-global-exemple'>
      <item action='deny' order='7'/>
    </list>
  </query>
</iq>
```

Par suite de la création et l'application de la liste précédente, l'usager ne va recevoir aucune communication de, ni envoyer aucune strophe à tous les autres usagers.

10.14 L'entité bloquée tente de communiquer avec l'usager

Si une entité bloquée tente d'envoyer des messages ou strophes de présence à l'usager, le serveur de l'usager DEVRAIT éliminer en silence la strophe et NE DOIT PAS retourner une erreur à l'entité envoyeuse.

Si une entité bloquée tente d'envoyer une strophe IQ de type "get" ou "set" à l'usager, le serveur de l'usager DOIT retourner à l'entité envoyeuse une erreur de strophe <service-unavailable/>, car c'est le code d'erreur standard envoyé d'un client qui ne comprend pas l'espace de noms d'une IQ get ou set. Les strophes IQ des autres types DEVRAIENT être éliminées en silence par le serveur.

Exemple : L'entité bloquée tente d'envoyer une IQ get :

```
<iq type='get'
    to='romeo@exemple.net'
    from='tybalt@exemple.com/pda'
    id='probing1'>
  <query xmlns='jabber:iq:version'/>
</iq>
```

Exemple : Le serveur retourne une erreur à l'entité bloquée :

```
<iq type='error'
    from='romeo@exemple.net'
    to='tybalt@exemple.com/pda'
```

```
   id='probing1'>
  <query xmlns='jabber:iq:version'/>
  <error type='cancel'>
    <service-unavailable
       xmlns='urn:ietf:params:xml:ns:xmpp-strophes'/>
  </error>
</iq>
```

10.15 Heuristique de haut niveau

Lors de la construction d'une représentation d'une heuristique de confidentialité de haut niveau, un client DEVRAIT utiliser la représentation la plus simple possible.

Par exemple, l'heuristique "bloquer toutes les communications avec tout usager qui n'est pas sur mon rôle" pourrait être construite d'une des façons suivantes :

- o permettre les communications provenant de tous les JID de mon rôle (c'est-à-dire, faire la liste de chaque JID comme un élément de liste séparé) mais bloquer les communications avec tous les autres ;
- o permettre les communications provenant de tout usager qui est dans un des groupes qui constituent mon rôle (c'est-à-dire, faire la liste de chaque groupe comme un élément de liste séparé) mais bloquer les communications provenant de tous les autres ;
- o permettre les communications provenant de tout usager avec lequel j'ai un abonnement de 'both' ou 'to' ou 'from' (c'est-à-dire, faire la liste de chaque valeur d'abonnement séparément) mais bloquer les communications avec tous les autres ;
- o bloquer les communications de quiconque a une valeur d'état d'abonnement de 'none'.

La représentation finale est la plus simple et DEVRAIT être utilisée ; voici le XML qui serait envoyé dans ce cas :

```
<iq type='set' id='heuristic1'>
  <query xmlns='jabber:iq:privacy'>
    <list name='heuristic-exemple'>
      <item type='subscription'
            value='none'
            action='deny'
            order='437'/>
    </list>
  </query>
</iq>
```

11. Règles pour le traitement des strophes XML par le serveur

Les règles de base d'acheminement et de livraison pour les serveurs sont définies dans la [RFC3920]. La présente section définit les règles supplémentaires pour les serveurs de messagerie instantanée et présence conformes à XMPP.

11.1 Strophes entrantes

Si le nom d'hôte de la portion identifiant de domaine du JID contenu dans l'attribut 'to' d'une strophe entrante correspond à un nom d'hôte du serveur lui-même et si le JID contenu dans l'attribut 'to' est de la forme <usager@exemple.com> ou <usager@exemple.com/ressource>, le serveur DOIT d'abord appliquer toutes les listes de confidentialité (Section 10) qui sont en vigueur, puis suivre les règles définies ci-dessous :

1. Si le JID est de la forme <usager@domaine/ressource> et si une ressource disponible correspond au JID complet, le serveur du receveur DOIT livrer la strophe à cette ressource.

2. Autrement si le JID est de la forme <usager@domaine> ou <usager@domaine/ ressource> et si le compte d'usager associé n'existe pas, le serveur du receveur (a) DEVRAIT ignorer en silence la strophe (c'est-à-dire, ni la livrer ni retourner une erreur) si c'est une strophe de présence, (b) DOIT retourner une erreur de strophe <service-unavailable/> à l'envoyeur si c'est une strophe IQ, et (c) DEVRAIT retourner une erreur de strophe <service-unavailable/> à l'envoyeur si c'est une strophe de message.

3. Autrement, si le JID est de la forme <usager@domaine/ressource> et si aucune ressource disponible ne correspond au JID complet, le serveur du receveur (a) DEVRAIT ignorer la strophe en silence (c'est-à-dire, ni la livrer ni retourner une erreur)

si c'est une strophe de présence, (b) DOIT retourner une erreur de strophe <service-unavailable/> à l'envoyeur si c'est une strophe IQ, et (c) DEVRAIT traiter la strophe comme si elle était adressée à <usager@domaine> si c'est une strophe de message.

4. Autrement, si le JID est de la forme <usager@domaine> et si il y a au moins une ressource disponible pour l'usager, le serveur du receveur DOIT suivre ces règles :

    1. Pour les strophes de message, le serveur DEVRAIT livrer la strophe à la ressource disponible de la plus forte priorité (si la ressource n'a pas fourni de valeur pour l'élément <priority/>, le serveur DEVRAIT considérer qu'elle a fourni une valeur de zéro). Si deux ressources disponibles ou plus ont la même priorité, le serveur PEUT utiliser une autre règle (par exemple, l'heure de connexion la plus récente, l'heure d'activité la plus récente, ou la plus forte disponibilité déterminée par une hiérarchie de valeurs de <show/>) pour choisir entre elles ou PEUT livrer le message à toutes ces ressources. Cependant, le serveur NE DOIT PAS livrer la strophe à une ressource disponible d'une priorité négative ; si la seule ressource disponible a une priorité négative, le serveur DEVRAIT traiter le message comme si il n'y avait pas de ressource disponible (défini ci-dessous). De plus, le serveur NE DOIT PAS réécrire l'attribut 'to' (c'est-à-dire, il DOIT le laisser comme <usager@domaine> plutôt que de le changer en <usager@domaine/ressource>).
    2. Pour les strophes de présence autres que celles de type "probe", le serveur DOIT livrer la strophe à toutes les ressources disponibles ; pour les sondes de présence, le serveur DEVRAIT répondre sur la base des règles définies au paragraphe 5.1.3 "Sondes de présence". De plus, le serveur NE DOIT PAS réécrire l'attribut 'to' (c'est-à-dire, il DOIT le laisser comme <usager@domaine> plutôt que le changer en <usager@domaine/ressource>).

    3. Pour les strophes IQ, le serveur DOIT lui-même répondre au nom de l'usager par une IQ de résultat ou d'erreur, et NE DOIT PAS livrer la strophe IQ à une des ressources disponibles. Précisément, si la sémantique de l'espace de noms qualifiant définit une réponse que le serveur peut fournir, le serveur DOIT répondre à la strophe au nom de l'usager ; sinon, le serveur DOIT répondre par une erreur de strophe <service-unavailable/>.

5. Autrement, si le JID est de la forme <usager@domaine> et si il n'y a pas de ressource disponible associée à l'usager, la façon dont la strophe est traitée dépend du type de la strophe :

    1. Pour les strophes de présence de type "subscribe", "subscribed", "unsubscribe", et "unsubscribed", le serveur DOIT tenir un enregistrement de la strophe et livrer la strophe au moins une fois (c'est-à-dire, la prochaine fois que l'usager crée une ressource disponible) ; de plus, le serveur DOIT continuer de livrer des strophes de présence de type "subscribe" jusqu'à ce que l'usager approuve ou refuse la demande d'abonnement (voir aussi au paragraphe 5.1.6 "Abonnements à présence").

    2. Pour toutes les autres strophes de présence, le serveur DEVRAIT ignorer en silence la strophe en ne la mémorisant pas pour livraison ultérieure ou en y répondant au nom de l'usager.

    3. Pour les strophes de message, le serveur PEUT choisir de mémoriser la strophe au nom de l'usager et de la livrer la prochaine fois que l'usager deviendra disponible, ou de transmettre le message à l'usager via quelque autre moyen (par exemple, au compte de messagerie électronique de l'usager). Cependant, si la mémorisation hors ligne du message ou sa transmission n'est pas activée, le serveur DOIT retourner une erreur de strophe <service-unavailable/> à l'envoyeur. (Noter que la mémorisation hors ligne et la transmission de message ne sont pas définies par XMPP, parce que c'est strictement une affaire de mise en œuvre et de provisionnement de service.)

    4. Pour les strophes IQ, le serveur DOIT répondre lui-même au nom de l'usager avec une IQ de résultat ou d'erreur. Précisément, si la sémantique de l'espace de noms qualifiant définit une réponse que peut fournir le serveur, celui-ci DOIT répondre à la strophe au nom de l'usager ; sinon, le serveur DOIT répondre par une erreur de strophe <service-unavailable/>.

11.2 Strophes sortantes

Si le nom d'hôte de la portion identifiant de domaine de l'adresse contenue dans l'attribut 'to' d'une strophe sortante correspond à un nom d'hôte du serveur lui-même, le serveur DOIT livrer la strophe à une entité locale conformément aux règles du paragraphe 11.1 "Strophes entrantes".

Si le nom d'hôte de la portion identifiant de domaine de l'adresse contenue dans l'attribut 'to' d'une strophe sortante ne correspond pas à un nom d'hôte du serveur lui-même, le serveur DOIT tenter d'acheminer la strophe au domaine étranger. L'ordre recommandé des actions est le suivant :

1. D'abord tenter de résoudre le nom d'hôte étranger en utilisant un service de la [RFC2782] de "xmpp-server" et un protocole de "tcp", résultant en enregistrements de ressource tels que "_xmpp-server._tcp.exemple.com.", comme spécifié dans la

[RFC3920].

2. Si la résolution de l'enregistrement d'adresse "xmpp-server" échoue, on tente de résoudre le service "_im" ou "_pres" de la [RFC2782] comme spécifié dans la [RFC3861], en utilisant le service "_im" pour les strophes <message/> et le service "_pres" pour les strophes <presence/> (il revient à la mise en œuvre de décider comment traiter les strophes <iq/>). Il va en résulter une ou plusieurs résolutions de la forme "_im.<proto>.exemple.com." ou "_pres.<proto>.exemple.com.", où "<proto>" sera une étiquette enregistrée dans le registre des étiquettes SRV de protocole de messagerie instantanée : soit "_xmpp" pour un domaine à capacité XMPP, soit quelque autre étiquette enregistrée par l'IANA (par exemple, "_simple") pour un domaine sans capacité XMPP.

3. Si les deux résolutions d'enregistrement d'adresse de SRV échouent, on tente d'effectuer une résolution normale d'enregistrement d'adresse IPv4/IPv6 pour déterminer l'adresse IP en utilisant l'accès "xmpp-server" de 5269 enregistré par l'IANA, comme spécifié dans la [RFC3920].

Les administrateurs de mises en œuvre de serveurs sont vivement encouragés à maintenir les enregistrements SRV _im._xmpp, _pres._xmpp, et _xmpp._tcp SRV correctement synchronisés, car des mises en œuvre différentes pourraient effectuer les recherches "_im" et "_pres" avant la recherche "xmpp-server".

12. Exigences de conformité pour IM et Présence

La présente section résume les aspects spécifiques du protocole extensible de messagerie instantanée et de présence qui DOIVENT être pris en charge par les serveurs et les clients de messagerie instantanée et de présence afin d'être considérés comme des mises en œuvre conformes. Toutes ces applications DOIVENT se conformer aux exigences spécifiées dans la [RFC3920]. Le texte de cette section spécifie des exigences de conformité supplémentaires pour les serveurs et clients de messagerie instantanée et présence ; noter bien que les exigences décrites ici complètent mais ne se substituent pas au cœur des exigences. Noter aussi qu'un serveur ou client PEUT ne prendre en charge que la présence ou que la messagerie instantanée, et n'est pas obligé de prendre en charge les deux si on désire seulement un service de présence ou un service de messagerie instantanée.

12.1 Serveurs

En plus des exigences centrales de conformité de serveur, un serveur de messagerie instantanée et présence DOIT en plus prendre en charge les protocoles suivants :

- toute la syntaxe et la sémantique de messagerie instantanée et présence relative au serveur définies dans le présent document, incluant la diffusion de présence au nom des clients, les abonnements à présence, la mémorisation et la manipulation de rôle, les listes de confidentialité, et les règles d'acheminement et de livraison spécifiques de la messagerie instantanée.

12.2 Clients

En plus des exigences centrales de conformité de client, un client de messagerie instantanée et présence DOIT en plus prendre en charge les protocoles suivants :

- génération et traitement de la sémantique spécifique d'IM des strophes XML comme défini par les schémas XML, incluant l'attribut 'type' de message et de strophe de présence ainsi que leurs éléments fils ;
- toute la syntaxe et sémantique de messagerie instantanée relative au client définies dans le présent document, incluant les abonnements à présence, la gestion de rôle, et les listes de confidentialité ;
- le chiffrement de bout en bout comme défini dans la [RFC3923] "Signature et chiffrement d'objet de bout en bout pour le protocole de messagerie et de présence extensibles (XMPP)".

Un client DOIT aussi traiter les adresses qui sont codées comme des URL "im:" comme spécifié dans la [RFC3860], et PEUT le faire en supprimant le schéma "im:" et en confiant la résolution d'adresse au serveur, comme spécifié au paragraphe 11.2 "Strophes sortantes".

13. Considérations d'internationalisation

Pour les considérations d'internationalisation, se reporter à la section pertinente de la [RFC3920].

14. Considérations sur la sécurité

Le cœur des considérations sur la sécurité de XMPP est défini dans la section pertinente de la [RFC3920].

Des considérations supplémentaires qui ne s'appliquent qu'aux applications de messagerie instantanée et de présence de XMPP sont définie en plusieurs endroits du présent mémoire ; précisément :

- Lorsque un serveur traite une strophe entrante de toute sorte dont le receveur désigné est un utilisateur associé à un des noms d'hôte du serveur, le serveur DOIT d'abord appliquer toutes les listes de confidentialité (Section 10) qui sont en vigueur (voir la Section 11 "Règles de traitement des strophes XML par le serveur").

- Lorsque un serveur traite une strophe de présence entrante de type "probe" dont le receveur désigné est un usager associé à un des noms d'hôte du serveur, le serveur NE DOIT PAS révéler les informations de présence de l'usager si l'envoyeur est une entité qui n'est pas autorisée à recevoir cette information comme déterminé par les abonnements de présence (voir le paragraphe 5.1 "Responsabilités du client et du serveur de présence").

- Lorsque un serveur traite une strophe de présence sortante sans type ou de type "unavailable", il DOIT suivre les règles définies au paragraphe 5.1 "Responsabilités du client et du serveur de présence" afin de s'assurer que de telles informations de présence ne sont pas diffusées à des entités qui ne sont pas autorisées à connaître de telles informations.

- Lorsque un serveur génère une strophe d'erreur en réponse à la réception d'une strophe pour un usager qui n'existe pas, l'utilisation de la condition d'erreur <service-unavailable/> aide à protéger contre les attaques bien connues de dictionnaire, car c'est la même condition d'erreur qui est retournée si, par exemple, l'espace de noms d'un élément fils d'IQ n'est pas compris, ou si une mémorisation hors ligne de message ou transmission de message n'est pas activée pour un domaine.

15. Considérations relatives à l'IANA

Pour certaines considérations relatives à l'IANA, se référer à la section pertinente de la [RFC3920].

15.1 Nom d'espace de noms XML pour les données de session

Un sous espace de noms d'URN pour les données relatives aux sessions dans le protocole extensible de messagerie instantanée et de présence (XMPP) est défini comme suit. (Ce nom d'espace de noms adhère au format défini dans le registre XML de l'IETF [RFC3688].)

URI : urn:ietf:params:xml:ns:xmpp-session
Spécification : RFC 3921
Description : c'est le nom d'espace de noms XML pour les données en rapport avec la session dans le protocole extensible de messagerie instantanée et de présence (XMPP) tel que défini dans la RFC 3921.
Contact d'enregistrement : IETF, groupe de travail XMPP, < xmppwg@jabber.org >

15.2 Enregistrement d'étiquettes de protocole de serveur de messagerie instantanée

La résolution d'adresse pour la messagerie instantanée et la présence [RFC3861] définit un registre d'étiquettes du protocole de SRV de messagerie instantanée pour les protocoles qui peuvent fournir des services qui se conforment à l'étiquette de service SRV "_im". Parce que XMPP est un de ces protocoles, l'IANA enregistre l'étiquette de protocole "_xmpp" dans le registre approprié comme suit :

Étiquette de protocole : _xmpp
Spécification : RFC 3921
Description : Étiquette de protocole de messagerie instantanée pour le protocole extensible de messagerie instantanée et de présence (XMPP) comme défini par la RFC 3921.
Contact d'enregistrement : IETF, groupe de travail XMPP, < xmppwg@jabber.org >

15.3 Enregistrement d'étiquettes de protocole de serveur de Présence

La résolution d'adresse pour la messagerie instantanée et la présence [RFC3861] définit un registre d'étiquettes du protocole de SRV de présence pour les protocoles qui peuvent fournir des services qui se conforment à l'étiquette de service SRV "_pres".

Parce que XMPP est un de ces protocoles, l'IANA enregistre l'étiquette de protocole "_xmpp" dans le registre approprié comme suit :

Étiquette de protocole : _xmpp
Spécification : RFC 3921
Description : Étiquette de protocole de présence pour le protocole extensible de messagerie instantanée et de présence (XMPP) comme défini par la RFC 3921.
Contact d'enregistrement : IETF, groupe de travail XMPP, < xmppwg@jabber.org >

16. Références

16.1 Références normatives

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

[RFC2779] M. Day et autres, "Exigences des protocoles Messagerie instantanée / Presence", février 2000. *(Information)*

[RFC2782] A. Gulbrandsen, P. Vixie et L. Esibov, "Enregistrement de ressource DNS pour la spécification de la localisation des services (DNS SRV)", février 2000.

[RFC3860] J. Peterson, "Profil commun pour la messagerie instantanée (CPIM)", août 2004. *(P.S.)*

[RFC3861] J. Peterson, "Résolution d'adresse pour la messagerie instantanée et les services de présence", août 2004. *(P.S.)*

[RFC3920] P. Saint-Andre, éd., "Protocole extensible de messagerie instantanée et de présence (XMPP) : éléments centraux", octobre 2004. *(P.S.)*

[RFC3923] P. Saint-Andre, "Signature et chiffrement d'objet de bout en bout pour le protocole de messagerie et de présence extensibles (XMPP)", octobre 2004. *(P.S.)*

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C., et E. Maler, "Extensible Markup Language (XML) 1.0 (2nd ed)", W3C REC-xml, octobre 2000, < http://pages.videotron.com/fyergeau/w3c/xml10/REC-xml-19980210.fr.html >.

[XML-NAMES] Bray, T., Hollander, D., et A. Layman, "Namespaces in XML", W3C REC-xml-names, janvier 1999, <http://www.w3.org/TR/REC-xml-names >.

16.2 Références pour information

[JEP-0054] Saint-Andre, P., "vcard-temp", JSF JEP 0054, mars 2003.

[JEP-0077] Saint-Andre, P., "In-Band Registration", JSF JEP 0077, août 2004.

[JEP-0078] Saint-Andre, P., "Non-SASL Authentication", JSF JEP 0078, juillet 2004.

[JSF] Jabber Software Foundation, "Jabber Software Foundation", < http://www.jabber.org/ >.

[RFC1459] J. Oikarinen et D. Reed, "Protocole Internet de relais de débats", mai 1993. *(Exp., MàJ par* 2810-13*)*

[RFC2426] F. Dawson, T. Howes, "Profil de répertoire MIME vCard", septembre 1998. *(Obsolète, voir* RFC6350*) (P.S.)*

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

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

Appendice A. vCards

Les paragraphes 3.1.3 et 4.1.4 de la [RFC2779] exigent qu'il soit possible de restituer des informations de contact hors bande pour les autres usagers (par exemple, un numéro de téléphone ou une adresse de messagerie électronique). Une représentation XML de la spécification vCard définie dans la [RFC2426] est d'usage courant dans la communauté Jabber pour fournir de

telles informations mais sort du domaine d'application de XMPP (la documentation de ce protocole est contenue dans [JEP-0054], publié par la Fondation des logiciels Jabber [JSF]).

Appendice B. Schémas XML

Les schémas XML suivants sont descriptifs, et non normatifs. Pour les schémas qui définissent les caractéristiques du cœur de XMPP, se reporter à la [RFC3920].

B.1 jabber:client

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

<xs:schema
   xmlns:xs='http://www.w3.org/2001/XMLSchema'
   targetNamespace='jabber:client'
   xmlns='jabber:client'
   elementFormDefault='qualified'>

 <xs:import namespace='urn:ietf:params:xml:ns:xmpp-strophes'/>

 <xs:element name='message'>
   <xs:complexType>
     <xs:sequence>
       <xs:choice minOccurs='0' maxOccurs='unbounded'>
         <xs:element ref='subject'/>
         <xs:element ref='body'/>
         <xs:element ref='thread'/>
       </xs:choice>
       <xs:any namespace='##other'
               minOccurs='0'
               maxOccurs='unbounded'/>
       <xs:element ref='error'
               minOccurs='0'/>
     </xs:sequence>
     <xs:attribute name='from'
               type='xs:string'
               use='optional'/>
     <xs:attribute name='id'
               type='xs:NMTOKEN'
               use='optional'/>
     <xs:attribute name='to'
               type='xs:string'
               use='optional'/>
     <xs:attribute name='type' use='optional' default='normal'>
       <xs:simpleType>
         <xs:restriction base='xs:NCName'>
           <xs:enumeration value='chat'/>
           <xs:enumeration value='error'/>
           <xs:enumeration value='groupchat'/>
           <xs:enumeration value='headline'/>
           <xs:enumeration value='normal'/>
         </xs:restriction>
       </xs:simpleType>
     </xs:attribute>
     <xs:attribute ref='xml:lang' use='optional'/>
   </xs:complexType>
 </xs:element>

 <xs:element name='body'>
   <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:element name='subject'>
  <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:element name='thread' type='xs:NMTOKEN'/>

<xs:element name='presence'>
  <xs:complexType>
    <xs:sequence>
     <xs:choice minOccurs='0' maxOccurs='unbounded'>
      <xs:element ref='show'/>
      <xs:element ref='status'/>
      <xs:element ref='priority'/>
     </xs:choice>
     <xs:any namespace='##other'
             minOccurs='0'
             maxOccurs='unbounded'/>
     <xs:element ref='error'
             minOccurs='0'/>
    </xs:sequence>
    <xs:attribute name='from'
             type='xs:string'
             use='optional'/>
    <xs:attribute name='id'
             type='xs:NMTOKEN'
             use='optional'/>
    <xs:attribute name='to'
             type='xs:string'
             use='optional'/>
    <xs:attribute name='type' use='optional'>
     <xs:simpleType>
      <xs:restriction base='xs:NCName'>
       <xs:enumeration value='error'/>
       <xs:enumeration value='probe'/>
       <xs:enumeration value='subscribe'/>
       <xs:enumeration value='subscribed'/>
       <xs:enumeration value='unavailable'/>
       <xs:enumeration value='unsubscribe'/>
       <xs:enumeration value='unsubscribed'/>
      </xs:restriction>
     </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref='xml:lang' use='optional'/>
  </xs:complexType>
</xs:element>

<xs:element name='show'>
  <xs:simpleType>
    <xs:restriction base='xs:NCName'>
```

```
      <xs:enumeration value='away'/>
      <xs:enumeration value='chat'/>
      <xs:enumeration value='dnd'/>
      <xs:enumeration value='xa'/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name='status'>
  <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:element name='priority' type='xs:byte'/>

<xs:element name='iq'>
  <xs:complexType>
    <xs:sequence>
      <xs:any namespace='##other'
              minOccurs='0'/>
      <xs:element ref='error'
              minOccurs='0'/>
    </xs:sequence>
    <xs:attribute name='from'
              type='xs:string'
              use='optional'/>
    <xs:attribute name='id'
              type='xs:NMTOKEN'
              use='required'/>
    <xs:attribute name='to'
              type='xs:string'
              use='optional'/>
    <xs:attribute name='type' use='required'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='error'/>
          <xs:enumeration value='get'/>
          <xs:enumeration value='result'/>
          <xs:enumeration value='set'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref='xml:lang' use='optional'/>
  </xs:complexType>
</xs:element>

<xs:element name='error'>
  <xs:complexType>
    <xs:sequence  xmlns:err='urn:ietf:params:xml:ns:xmpp-strophes'>
      <xs:group ref='err:stanzaErrorGroup'/>
      <xs:element ref='err:text' minOccurs='0'/>
    </xs:sequence>
    <xs:attribute name='code' type='xs:byte' use='optional'/>
    <xs:attribute name='type' use='required'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='auth'/>
          <xs:enumeration value='cancel'/>
```

```
          <xs:enumeration value='continue'/>
          <xs:enumeration value='modify'/>
          <xs:enumeration value='wait'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
 </xs:element>

</xs:schema>
```

B.2 jabber:server

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

<xs:schema
   xmlns:xs='http://www.w3.org/2001/XMLSchema'
   targetNamespace='jabber:server'
   xmlns='jabber:server'
   elementFormDefault='qualified'>

 <xs:import namespace='urn:ietf:params:xml:ns:xmpp-strophes'/>

 <xs:element name='message'>
   <xs:complexType>
     <xs:sequence>
       <xs:choice minOccurs='0' maxOccurs='unbounded'>
         <xs:element ref='subject'/>
         <xs:element ref='body'/>
         <xs:element ref='thread'/>
       </xs:choice>
       <xs:any namespace='##other'
               minOccurs='0'
               maxOccurs='unbounded'/>
       <xs:element ref='error'
               minOccurs='0'/>
     </xs:sequence>
     <xs:attribute name='from'
               type='xs:string'
               use='required'/>
     <xs:attribute name='id'
               type='xs:NMTOKEN'
               use='optional'/>
     <xs:attribute name='to'
               type='xs:string'
               use='required'/>
     <xs:attribute name='type' use='optional' default='normal'>
       <xs:simpleType>
         <xs:restriction base='xs:NCName'>
           <xs:enumeration value='chat'/>
           <xs:enumeration value='error'/>
           <xs:enumeration value='groupchat'/>
           <xs:enumeration value='headline'/>
           <xs:enumeration value='normal'/>
         </xs:restriction>
       </xs:simpleType>
     </xs:attribute>
     <xs:attribute ref='xml:lang' use='optional'/>
   </xs:complexType>
 </xs:element>

 <xs:element name='body'>
```

```
  <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:element name='subject'>
  <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:element name='thread' type='xs:NMTOKEN'/>

 <xs:element name='presence'>
  <xs:complexType>
   <xs:sequence>
    <xs:choice minOccurs='0' maxOccurs='unbounded'>
     <xs:element ref='show'/>
     <xs:element ref='status'/>
     <xs:element ref='priority'/>
    </xs:choice>
    <xs:any  namespace='##other'
             minOccurs='0'
             maxOccurs='unbounded'/>
    <xs:element ref='error'
             minOccurs='0'/>
   </xs:sequence>
   <xs:attribute name='from'
             type='xs:string'
             use='required'/>
   <xs:attribute name='id'
             type='xs:NMTOKEN'
             use='optional'/>
   <xs:attribute name='to'
             type='xs:string'
             use='required'/>
   <xs:attribute name='type' use='optional'>
    <xs:simpleType>
     <xs:restriction base='xs:NCName'>
      <xs:enumeration value='error'/>
      <xs:enumeration value='probe'/>
      <xs:enumeration value='subscribe'/>
      <xs:enumeration value='subscribed'/>
      <xs:enumeration value='unavailable'/>
      <xs:enumeration value='unsubscribe'/>
      <xs:enumeration value='unsubscribed'/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute ref='xml:lang' use='optional'/>
  </xs:complexType>
 </xs:element>

 <xs:element name='show'>
  <xs:simpleType>
```

```
    <xs:restriction base='xs:NCName'>
      <xs:enumeration value='away'/>
      <xs:enumeration value='chat'/>
      <xs:enumeration value='dnd'/>
      <xs:enumeration value='xa'/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name='status'>
  <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:element name='priority' type='xs:byte'/>

<xs:element name='iq'>
  <xs:complexType>
    <xs:sequence>
      <xs:any namespace='##other'
              minOccurs='0'/>
      <xs:element ref='error'
              minOccurs='0'/>
    </xs:sequence>
    <xs:attribute name='from'
              type='xs:string'
              use='required'/>
    <xs:attribute name='id'
              type='xs:NMTOKEN'
              use='required'/>
    <xs:attribute name='to'
              type='xs:string'
              use='required'/>
    <xs:attribute name='type' use='required'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='error'/>
          <xs:enumeration value='get'/>
          <xs:enumeration value='result'/>
          <xs:enumeration value='set'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute ref='xml:lang' use='optional'/>
  </xs:complexType>
</xs:element>

<xs:element name='error'>
  <xs:complexType>
    <xs:sequence  xmlns:err='urn:ietf:params:xml:ns:xmpp-strophes'>
      <xs:group   ref='err:stanzaErrorGroup'/>
      <xs:element ref='err:text'
              minOccurs='0'/>
    </xs:sequence>
    <xs:attribute name='code' type='xs:byte' use='optional'/>
    <xs:attribute name='type' use='required'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='auth'/>
```

```
          <xs:enumeration value='cancel'/>
          <xs:enumeration value='continue'/>
          <xs:enumeration value='modify'/>
          <xs:enumeration value='wait'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
 </xs:element>

</xs:schema>
```

B.3 session

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-session'
    xmlns='urn:ietf:params:xml:ns:xmpp-session'
    elementFormDefault='qualified'>

  <xs:element name='session' type='empty'/>

   <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

B.4 jabber:iq:privacy

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='jabber:iq:privacy'
    xmlns='jabber:iq:privacy'
    elementFormDefault='qualified'>

  <xs:element name='query'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='active'
                    minOccurs='0'/>
        <xs:element ref='default'
                    minOccurs='0'/>
        <xs:element ref='list'
                    minOccurs='0'
                    maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='active'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:NMTOKEN'>
          <xs:attribute name='name'
```

```
                type='xs:string'
                use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name='default'>
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base='xs:NMTOKEN'>
        <xs:attribute name='name'
                type='xs:string'
                use='optional'/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name='list'>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='item'
                minOccurs='0'
                maxOccurs='unbounded'/>
    </xs:sequence>
    <xs:attribute name='name'
                type='xs:string'
                use='required'/>
  </xs:complexType>
</xs:element>

<xs:element name='item'>
  <xs:complexType>
    <xs:sequence>
      <xs:element name='iq'
                minOccurs='0'
                type='empty'/>
      <xs:element name='message'
                minOccurs='0'
                type='empty'/>
      <xs:element name='presence-in'
                minOccurs='0'
                type='empty'/>
      <xs:element name='presence-out'
                minOccurs='0'
                type='empty'/>
    </xs:sequence>
    <xs:attribute name='action' use='required'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='allow'/>
          <xs:enumeration value='deny'/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name='order'
                type='xs:unsignedInt'
                use='required'/>
    <xs:attribute name='type' use='optional'>
      <xs:simpleType>
        <xs:restriction base='xs:NCName'>
          <xs:enumeration value='group'/>
```

```
          <xs:enumeration value='jid'/>
          <xs:enumeration value='subscription'/>
         </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name='value'
                type='xs:string'
                use='optional'/>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

B.5 jabber:iq:roster

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='jabber:iq:roster'
    xmlns='jabber:iq:roster'
    elementFormDefault='qualified'>

  <xs:element name='query'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='item'
                minOccurs='0'
                maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='item'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='group'
                minOccurs='0'
                maxOccurs='unbounded'/>
      </xs:sequence>
      <xs:attribute name='ask' use='optional'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='subscribe'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name='jid' type='xs:string' use='required'/>
      <xs:attribute name='name' type='xs:string' use='optional'/>
      <xs:attribute name='subscription' use='optional'>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='both'/>
            <xs:enumeration value='from'/>
            <xs:enumeration value='none'/>
            <xs:enumeration value='remove'/>
            <xs:enumeration value='to'/>
```

```
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name='group' type='xs:string'/>

</xs:schema>
```

Appendice C. Différences entre les protocoles Jabber IM/Presence et XMPP

Cet appendice n'est pas normatif.

XMPP a été adapté des protocoles développés à l'origine dans la communauté libre Jabber, qui peuvent être vus comme "XMPP 0.9". Comme il existe une large base installée de mises en œuvre et déploiements de Jabber, il peut être utile de spécifier les différences clés entre les protocoles Jabber pertinents et XMPP afin d'accélérer et encourager la mise à niveau de ces mises en œuvre et déploiements en XMPP. Le présent appendice résume les différences qui se rapportent spécifiquement aux applications de messagerie instantanée et de présence, tandis que la section correspondante de la [RFC3920] résume les différences qui se rapportent à toutes les applications XMPP.

C.1 Établissement de session

Le protocole d'authentification de client à serveur développé dans la communauté Jabber supposait que tout client est un client d'IM et initiait une session IM dès la réussite de l'authentification et du lien de ressource, qui sont effectuées simultanément (la documentation de ce protocole est contenue dans [JEP-0078], publié par la Fondation des logiciels Jabber [JSF]). XMPP établit une séparation plus stricte entre les fonctionnalités centrales et la fonction d'IM ; donc, une session d'IM n'est pas créée tant que le client n'en demande pas spécifiquement une en utilisant le protocole défini à la Section 3 "Établissement de session".

C.2 Listes de confidentialité

La communauté Jabber a commencé à définir un protocole pour bloquer les communications (listes de confidentialité) à la fin 2001, mais cet effort a été déconseillé après la formation du groupe de travail XMPP. Donc, le protocole défini à la Section 10 "Blocage de communication" est le seul protocole défini pour l'usage de la communauté Jabber.

Contributeurs

La plupart des aspects centraux du protocole extensible de messagerie instantanée et de présence ont été développés à l'origine au sein de la communauté libre Jabber en 1999. Cette communauté a été fondée par Jeremie Miller, qui a produit le code source pour la version initiale du serveur jabberd en janvier 1999. Les contributeurs majeurs précoces au protocole de base incluaient aussi Ryan Eatmon, Peter Millard, Thomas Muldowney, et Dave Smith. Les travaux spécifiques de la messagerie instantanée et présence par le groupe de travail XMPP se sont concentrés spécialement sur l'établissement de session d'IM et le blocage de communication (listes de confidentialité) ; le protocole d'établissement de session a principalement été développé par Rob Norris et Joe Hildebrand, et le protocole de listes de confidentialité a été fourni à l'origine par Peter Millard.

Remerciements

Des remerciements sont dus à un grand nombre de gens en plus des contributeurs cités. Bien qu'il soit difficile d'en fournir une liste complète, les personnes suivantes ont été d'une aide particulière pour définir les protocoles ou en commentant les spécifications du présent mémoire : Thomas Charron, Richard Dobson, Schuyler Heath, Jonathan Hogg, Craig Kaes, Jacek Konieczny, Lisa Dusseault, Alexey Melnikov, Keith Minkler, Julian Missig, Pete Resnick, Marshall Rose, Jean-Louis Seguineau, Alexey Shchepin, Iain Shigeoka, et David Waite. Merci aussi aux membres du groupe de travail XMPP et à la communauté de l'IETF pour les commentaires et retours fournis durant toute la vie de ce mémoire.

Adresse de l'auteur

Peter Saint-Andre (editor)

Jabber Software Foundation

mél : stpeter@jabber.org

Déclaration complète de droits de reproduction

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