Groupe de travail Réseau
Request for Comments : 3920
Catégorie : En cours de normalisation

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

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

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 définit les caractéristiques centrales du protocole extensible de messagerie instantanée et de présence (XMPP, *Extensible Messaging and Presence Protocol*), protocole pour écouler des éléments de langage de balisage extensible (XML, *Extensible Markup Language*) afin d'échanger des informations structurées en temps presque réel entre deux points d'extrémité quelconques de réseau. Bien que XMPP fournisse un cadre généralisé extensible pour l'échange de données XML, il est utilisé principalement pour les besoins de la construction d'applications de messagerie instantanée et de présence qui satisfont aux exigences de 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 libre en langage de balisage extensible (XML, *Extensible Markup Language*) [XML] pour des services de messagerie presque en temps réel, de présence, et d'interrogation-réponse. La syntaxe et la sémantique de base ont été développées à l'origine au sein de la communauté de logiciels libres Jabber, principalement en 1999. En 2002, le groupe de travail XMPP a été mandaté par l'IETF pour développer une adaptation du protocole Jabber qui convienne pour les technologies de messagerie instantanée (IM, *instant messaging*) et de présence, de l'IETF. Par suite des travaux du groupe XMPP, le présent mémoire définit les caractéristiques centrales de XMPP 1.0 ; les extensions nécessaires pour fournir les fonctionnalités de messagerie instantanée et de présence définies dans la [RFC2779] sont spécifiées dans la [RFC3921] "Protocole extensible de messagerie et de présence (XMPP) : messagerie instantanée et présence".

1.2 Terminologie

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

2. Architecture générale

2.1 Vue d'ensemble

Bien que XMPP ne soit lié à aucune architecture de réseau spécifique, il a été généralement mis en œuvre via une architecture client-serveur dans laquelle un client qui utilise XMPP accède à un serveur sur une connexion de la [RFC0793], et les serveurs communiquent aussi les uns avec les autres sur des connexions TCP.

Le diagramme suivant donne une vue très globale de cette architecture (où "-" représente les communications qui utilisent XMPP et "=" représente les communications qui utilisent tout autre protocole).

```
C1----S1---S2---C3
       |
C2----+-G1===FN1===FC1
```

Les symboles sont les suivants :
C1, C2, C3 = clients XMPP

S1, S2 = serveurs XMPP
G1 = une passerelle qui traduit entre XMPP et le ou les protocoles utilisés sur un réseau de messagerie étranger (non XMPP)
FN1 = réseau de messagerie étranger
FC1 = client sur un réseau de messagerie étranger

2.2 Serveur

Un serveur agit comme une couche d'abstraction intelligente pour les communications XMPP. Sa principale responsabilité est :
- o de gérer les connexions ou sessions pour les autres entités, sous la forme de flux XML (Section 4) de et vers des clients autorisés, des serveurs, et d'autres entités,
- o d'acheminer les strophes XML d'adresse appropriée (Section 9) entre de telles entités sur les flux XML.

La plupart des serveurs conformes à XMPP assument aussi la responsabilité de la mémorisation des données qui sont utilisées par les clients (par exemple, les listes de contacts des utilisateurs d'applications de messagerie instantanée et de présence fondées sur XMPP) ; dans ce cas, les données XML sont traitées directement par le serveur lui même au nom du client et ne sont pas acheminées par une autre entité.

2.3 Client

La plupart des clients se connectent directement à un serveur sur une connexion de la [RFC0793] et utilisent XMPP pour tirer pleinement parti de la fonctionnalité fournie par un serveur et de tous les services associés. Plusieurs ressources (par exemple, appareils ou localisations) PEUVENT se connecter simultanément à un serveur au nom de chaque client autorisé, avec chaque ressource différenciée par l'identifiant de ressource d'une adresse XMPP (par exemple, <nœud@domaine/domicile> vs. <nœud@domaine/bureau>) comme défini dans le schéma d'adressage (Section 3). L'accès RECOMMANDÉ pour les connexions entre un client et un serveur est 5222, qui est enregistré par l'IANA (voir au paragraphe 15.9 "Numéros d'accès").

2.4 Passerelle

Une passerelle est un service côté serveur d'un objet particulier dont la fonction principale est de traduire XMPP dans le protocole utilisé par un système étranger (non XMPP) de messagerie, ainsi que de traduire les données de retour en XMPP. Des exemples sont les passerelles avec la messagerie électronique (voir la [RFC2821]), les relais de débats sur Internet (voir la [RFC1459]), SIMPLE (voir [SIMPLE]), le service de messages courts (SMS, *Short Message Service*), et les services traditionnels de messagerie instantanée comme AIM, ICQ, MSN Messenger, et Yahoo! Instant Messenger. Les communications entre passerelles et serveurs, et entre passerelles et le système étranger de messagerie, ne sont pas définies dans le présent document.

2.5 Réseau

Parce que chaque serveur est identifié par une adresse réseau et parce que les communications de serveur à serveur sont une extension directe du protocole client-serveur, en pratique, le système consiste en un réseau de serveurs qui inter-communiquent. Donc, par exemple, <juliet@exemple.com> est capable d'échanger des messages, des informations de présence, et d'autres informations avec <romeo@exemple.net>. Ce schéma est familier des protocoles de messagerie (comme la [RFC2821]) qui font usage des normes d'adressage. Les communications entre deux serveurs quelconques sont FACULTATIVES. Si elles sont activées, de telles communications DEVRAIENT se faire sur des flux XML qui sont liés aux connexions de la [RFC0793]. L'accès RECOMMANDÉ pour les connexions entre les serveurs est 5269, comme enregistré par l'IANA (voir au paragraphe 15.9 "Numéros d'accès").

3. Schéma d'adressage

3.1 Vue d'ensemble

Une entité est toute chose qui peut être considérée comme point d'extrémité de réseau (c'est-à-dire, un identifiant sur le réseau) et qui peut communiquer en utilisant XMPP. Toutes ces entités sont adressables de façon univoque sous une forme qui est cohérente avec la [RFC2396]. Pour des raisons historiques, l'adresse d'une entité XMPP est appelée un identifiant Jabber, ou JID. Un JID valide contient un ensemble d'éléments ordonnés formé d'un identifiant de domaine, d'un identifiant de nœud, et d'un identifiant de ressource.

La syntaxe d'un JID est définie ci-dessous en utilisant la forme Backus-Naur augmenté définie dans la [RFC2234]. (Les règles de adresseIPv4 et adresseIPv6 sont définies à l'Appendice B de la [RFC3513] ; les séquences de caractères admises qui se

conforment à la règle 'nœud' sont définies par le profil Nodeprep de la [RFC3454] comme documenté à l'Appendice A du présent mémoire ; les séquences de caractères admises qui se conforment à la règle 'ressource' sont définies par le profil Resourceprep de la [RFC3454] comme documenté à l'Appendice B du présent mémoire ; et la règle 'sous-domaine' fait référence au concept d'étiquette de domaine internationalisée décrite dans la [RFC3490].)

```
jid              = [ nœud "@" ] domaine [ "/" ressource ]
domaine          = fqdn / adresse-littérale
fqdn             = (sous-domaine 1*("." sous-domaine))
sous-domaine     = (étiquette de domaine internationalisée)
adresse-littérale = adresseIPv4/ adresseIPv6
```

Tous les JID se fondent sur la structure précédente. L'utilisation la plus courante de cette structure est d'identifier un utilisateur de messagerie instantanée, le serveur auquel l'utilisateur se connecte, et la ressource connectée de l'utilisateur (par exemple, un client spécifique) sous la forme de <usager@hôte/ressource>. Cependant, les types de nœuds autres que des clients sont possibles ; par exemple, une salle de débat spécifique offerte par un service de débat multi-utilisateurs pourrait être située à <salle@service> (où "salle" est le nom de la salle de débat et "service" est le nom d'hôte du service de débat multi-utilisateurs) et un occupant spécifique d'une telle salle pourrait être situé à <salle@service/nick> (où "nick" est le surnom de la salle de l'occupant). De nombreux autres types de JID sont possibles (par exemple, <domaine/ressource> pourrait être un script ou service côté serveur).

Une portion admissible d'un JID (identifiant de nœud, identifiant de domaine, et identifiant de ressource) NE DOIT PAS faire plus de 1023 octets, résultant en une taille totale maximum (incluant le '@' et les séparateurs '/') de 3071 octets.

3.2 Identifiant de domaine

L'identifiant de domaine est le principal identifiant et est le seul élément EXIGÉ d'un JID (un simple identifiant de domaine est un JID valide). Il représente généralement la passerelle réseau ou serveur "principal" auquel les autres entités se connectent pour l'acheminement XML et les capacités de gestion des données. Cependant, l'entité référencée par un identifiant de domaine n'est pas toujours un serveur, et peut être un service auquel on s'adresse comme à un sous domaine d'un serveur qui fournit les fonctions ci-dessus et au delà des capacités d'un serveur (par exemple, un service de débat multi-utilisateurs, un répertoire d'utilisateurs, ou une passerelle vers un système de messagerie étranger).

L'identifiant de domaine pour chaque serveur ou service qui va communiquer sur un réseau PEUT être une adresse IP mais DEVRAIT être un nom de domaine pleinement qualifié (voir la [[RFC1035]). Un identifiant de domaine DOIT être un "nom de domaine internationalisé" comme défini dans la [RFC3490], auquel le profil Nameprep [RFC3491] de stringprep [RFC3454] peut être appliqué sans échec. Avant de comparer deux identifiants de domaine, un serveur DOIT (et un client DEVRAIT) d'abord appliquer le profil Nameprep aux étiquettes (comme défini dans la [RFC3490]) qui constituent chaque identifiant.

3.3 Identifiant de nœud

L'identifiant de nœud est un identifiant secondaire facultatif placé avant l'identifiant de domaine et séparé de lui par le caractère '@'. Il représente généralement l'entité demandeuse et utilise un accès réseau fourni par le serveur ou la passerelle (c'est-à-dire, un client) bien qu'il puisse aussi représenter d'autres sortes d'entités (par exemple, une salle de débat associée à un service de débat multi-utilisateur). L'entité représentée par un identifiant de nœud est localisée dans le contexte d'un domaine spécifique ; dans les applications de messagerie instantanée et de présence de XMPP, cette adresse est appelée un "JID nu" et elle est de la forme <nœud@domaine>.

Un identifiant de nœud DOIT être formaté de telle sorte que le profil Nodeprep de la [RFC3454] puisse lui être appliqué sans échec. Avant de comparer deux identifiants de nœud, un serveur DOIT (et un client DEVRAIT) d'abord appliquer le profil Nodeprep à chaque identifiant.

3.4 Identifiant de ressource

L'identifiant de ressource est un identifiant facultatif tertiaire placé après l'identifiant de domaine et séparé de lui par le caractère '/'. Un identifiant de ressource peut modifier un <nœud@domaine> ou une simple adresse <domaine>. Il représente généralement une session spécifique, une connexion (par exemple, un appareil ou une localisation) ou un objet (par exemple, un participant dans une salle de débat multi-utilisateurs) appartenant à l'entité associée à un identifiant de nœud. Un identifiant de ressource est opaque pour les serveurs et les autres clients, et est normalement défini par une mise en œuvre de client lorsque il fournit les informations nécessaires pour réaliser le lien de ressource (Section 7) (bien qu'il puisse être généré par un serveur au nom d'un client) d'après lequel on s'y réfère comme à une "ressource connectée". Une entité PEUT conserver

simultanément plusieurs ressources connectées, dont chacune est différenciée par un identifiant de ressource distinct.

Un identifiant de ressource DOIT être formé de telle façon que le profil Resourceprep de la [RFC3454] puisse être appliqué sans échec. Avant de comparer deux identifiants de ressource, un serveur DOIT (et un client DEVRAIT) d'abord appliquer le profil Resourceprep à chaque identifiant.

3.5 Détermination des adresses

Après la négociation SASL (Section 6) et, si approprié, le lien de ressource (Section 7) l'entité receveuse pour un flux DOIT déterminer le JID de l'entité initiatrice.

Pour les communications de serveur à serveur, le JID de l'entité initiatrice DEVRAIT être l'identité d'autorisation, déduite de l'identité d'authentification, comme défini par la spécification du protocole d'authentification simple et couche de sécurité (SASL, *Simple Authentication et Security Layer*) [RFC2222], si aucune identité d'autorisation n'a été spécifiée durant la négociation SASL (Section 6).

Pour les communications de client à serveur, le "JID nu" (<nœud@domaine>) DEVRAIT être l'identité d'autorisation, déduite de l'identité d'authentification, comme définie dans la [RFC2222], si aucune identité d'autorisation n'était spécifiée durant la négociation SASL (Section 6) ; la portion identifiant de ressource du "JID complet" (<nœud@domaine/ressource>) DEVRAIT être l'identifiant de ressource négocié par le client et le serveur durant le lien de ressource (Section 7).

L'entité receveuse DOIT s'assurer que le JID résultant (incluant l'identifiant de nœud, l'identifiant de domaine, l'identifiant de ressource, et les caractères séparateurs) se conforme aux règles et formats définis plus haut dans cette section ; pour respecter cette restriction, l'entité receveuse peut avoir besoin de remplacer le JID envoyé par l'entité initiatrice avec le JID canonisé comme déterminé par l'entité receveuse.

4. Flux XML

4.1 Vue d'ensemble

Deux concepts fondamentaux rendent possible l'échange asynchrone rapide de charges utiles relativement petites d'informations structurées entre entités à capacité de présence : flux XML et strophes XML. Ces termes sont définis de la façon suivante :

Flux XML : un flux XML est un conteneur pour l'échange d'éléments XML entre deux entités quelconques sur un réseau. Le début d'un flux XML est noté sans ambiguïté par une étiquette d'ouverture XML <stream> (avec les déclarations appropriées d'attributs et d'espace de noms) tandis que la fin du flux XML est notée sans ambiguïté par une étiquette XML de fermeture </stream>. Durant la vie du flux, l'entité qui l'a initié peut envoyer un nombre non limité d'éléments XML sur le flux, soit des éléments utilisés pour négocier le flux (par exemple, pour négocier l'utilisation de TLS (Section 5) ou l'utilisation de SASL (Section 6)) soit des strophes XML (comme défini ci-dessous, éléments <message/>, <presence/>, ou <iq/> qualifiés par l'espace de noms par défaut). Le "flux initial" est négocié à partir de l'entité initiatrice (usuellement un client ou serveur) vers l'entité receveuse (usuellement un serveur) et peut être vu comme correspondant à la "session" de l'entité initiatrice avec l'entité receveuse. Le flux initial permet une communication unidirectionnelle de l'entité initiatrice à l'entité receveuse ; afin de permettre l'échange d'informations de l'entité receveuse à l'entité initiatrice, l'entité receveuse DOIT négocier un flux dans la direction opposée (le "flux de réponse").

Strophe XML : une strophe XML est une unité discrète de sémantique d'informations structurées qui est envoyée d'une entité à une autre sur un flux XML. Une strophe XML existe au niveau fils direct de l'élément racine <stream/> et est dite être bien équilibrée si elle correspond au contenu de production [43] de [XML]. Le début de toute strophe XML est noté de façon non ambiguë par l'étiquette de début d'élément à la profondeur=1 du flux XML (par exemple, <presence>) et la fin de toute strophe XML est noté sans ambiguïté par l'étiquette de fermeture correspondante à la profondeur=1 (par exemple, </presence>). Une strophe XML PEUT contenir des éléments fils (avec les attributs, éléments, et données de caractères XML correspondants) comme nécessaire afin de porter les informations désirées. Les seules strophes XML définies ici sont les éléments <message/>, <presence/>, et <iq/> qualifiés par l'espace de noms par défaut pour le flux, comme décrit sous "Strophes XML" (Section 9) ; un élément XML envoyé pour les besoins de la négociation de la sécurité de la couche transport (TLS, *Transport Layer Security*) (Section 5), la négociation de l'authentification simple et couche de sécurité (SASL, *Simple Authentication and Security Layer*) (Section 6), ou le rappel de serveur (Section 8) n'est pas considéré comme étant une strophe XML.

Considérons l'exemple d'une session d'un client avec un serveur. Afin de se connecter à un serveur, un client DOIT initier un flux XML par l'envoi au serveur d'une étiquette d'ouverture <stream>, facultativement précédée d'une déclaration textuelle qui

spécifie la version XML et le codage de caractères pris en charge (voir le paragraphe 11.4 "Inclusion d'une déclaration textuelle") ; voir aussi au paragraphe 11.5 "Codage de caractères"). Sous réserve des politiques locales et du provisionnement des services, le serveur DEVRAIT alors répondre au client par un second flux XML, là encore précédé facultativement par une déclaration textuelle. Une fois que le client a achevé la négociation SASL (Section 6), le client PEUT envoyer un nombre illimité de strophes XML sur le flux à tout receveur sur le réseau. Lorsque le client désire clore le flux, il envoie simplement une étiquette de fermeture </stream> au serveur (autrement, le flux peut être fermé par le serveur) après quoi le client et le serveur DEVRAIENT tous deux terminer aussi la connexion sous-jacente (généralement une connexion TCP).

Ceux qui ont l'habitude de voir XML comme centré sur le document peuvent souhaiter voir une session de client avec un serveur comme consistant en deux documents XML ouverts : un du client au serveur et un du serveur au client. Dans cette perspective, l'élément racine <stream/> peut être considéré comme l'entité document pour chaque "document", et les deux "documents" sont construits par l'accumulation des strophes XML envoyées sur les deux flux XML. Cependant, cette perspective est seulement une convention ; XMPP ne traite pas de documents mais de flux XML et de strophes XML.

Par essence, un flux XML agit donc comme une enveloppe pour toutes les strophes XML envoyées durant une session. On peut représenter cela de façon simpliste comme suit :

```
|--------------------|
| <stream>           |
|--------------------|
| <presence>         |
|   <show/>          |
| </presence>        |
|--------------------|
| <message to='foo'> |
|   <body/>          |
| </message>         |
|--------------------|
| <iq to='bar'>      |
|   <query/>         |
| </iq>              |
|--------------------|
| ...                |
|--------------------|
| </stream>          |
|--------------------|
```

4.2 Lien avec TCP

Bien qu'il n'y ait pas de couplage nécessaire d'un flux XML à une connexion de la [RFC0793] (par exemple, deux entités pourraient se connecter l'une à l'autre via un autre mécanisme comme une consultation sur hypertexte [RFC2616]), la présente spécification définit un lien de XMPP à TCP seulement. Dans le contexte des communications de client à serveur, un serveur DOIT permettre à un client de partager une seule connexion TCP pour les strophes XML envoyées de client à serveur et de serveur à client. Dans le contexte de communications de serveur à serveur, un serveur DOIT utiliser une connexion TCP pour les strophes XML envoyées du serveur à l'homologue et une autre connexion TCP (initiée par l'homologue) pour les strophes de l'homologue au serveur, pour un total de deux connexions TCP.

4.3 Sécurité de flux

Lors de la négociation de flux XML dans XMPP 1.0, TLS DEVRAIT être utilisé comme défini dans "Utilisation de TLS" (Section 5) et SASL DOIT être utilisé comme défini dans "Utilisation de SASL" (Section 6). Le "flux initial" (c'est-à-dire, le flux provenant de l'entité initiatrice pour l'entité receveuse) et le "flux de réponse" (c'est-à-dire, le flux de l'entité receveuse à l'entité initiatrice) DOIT être sécurisé séparément, bien que la sécurité dans les deux directions PUISSE être établie via des mécanismes qui assurent l'authentification mutuelle. Une entité NE DEVRAIT PAS tenter d'envoyer des strophes XML (Section 9) sur le flux avant que celui-ci soit authentifié, mais si il le fait, l'autre entité NE DOIT alors PAS accepter de telles strophes et DEVRAIT retourner une erreur de flux <non-autorisé/> et terminer le flux XML et la connexion TCP sous-jacente ; noter que ceci s'applique seulement aux strophes XML (c'est-à-dire aux éléments <message/>, <presence/>, et <iq/> de l'espace de noms par défaut) et non aux éléments XML utilisés pour la négociation de flux (par exemple, les éléments utilisés pour négocier l'utilisation de TLS (Section 5) ou l'utilisation de SASL (Section 6)).

4.4 Attributs de flux

Les attributs de l'élément flux sont les suivants :

to : l'attribut 'to' DEVRAIT être utilisé seulement dans l'en-tête de flux XML provenant de l'entité initiatrice à l'entité receveuse, et DOIT être réglé à un nom d'hôte desservi par l'entité receveuse. Il NE DEVRAIT PAS y avoir d'attribut 'to' établi dans l'en-tête de flux XML par lequel l'entité receveuse répond à l'entité initiatrice ; cependant, si un attribut 'to' est inclus, il DEVRAIT être ignoré en silence par l'entité initiatrice.

from : l'attribut 'from' DEVRAIT être utilisé seulement dans l'en-tête de flux XML provenant de l'entité receveuse à l'entité initiatrice, et DOIT être réglé à un nom d'hôte desservi par l'entité receveuse qui accorde l'accès à l'entité initiatrice. Il NE DEVRAIT PAS y avoir d'attribut 'from' dans l'en-tête de flux XML envoyé de l'entité initiatrice à l'entité receveuse ; cependant, si un attribut 'from' est inclus, il DEVRAIT être ignoré en silence par l'entité receveuse.

id : l'attribut 'id' DEVRAIT n'être utilisé que dans l'en-tête de flux XML de l'entité receveuse à l'entité initiatrice. Cet attribut est un identifiant unique créé par l'entité receveuse pour fonctionner comme une clé de session pour le flux de l'entité initiatrice avec l'entité receveuse, et DOIT être unique au sein de l'application receveuse (normalement un serveur). Noter que l'identifiant de flux peut être critique pour la sécurité et DOIT donc être à la fois imprévisible et non répétitif (voir dans la [RFC1750] les recommandations concernant l'aléa pour les besoins de sécurité). Il NE DEVRAIT PAS y avoir d'attribut 'id' dans l'en-tête de flux XML envoyé de l'entité initiatrice à l'entité receveuse ; cependant, si un attribut 'id' est inclus, il DEVRAIT être ignoré en silence par l'entité receveuse.

xml:lang : un attribut 'xml:lang' (comme défini au paragraphe 2.12 de [XML]) DEVRAIT être inclus par l'entité initiatrice dans l'en-tête du flux initial pour spécifier le langage par défaut de toutes données de caractères XML lisibles par l'homme qu'elle envoie sur ce flux. Si l'attribut est inclus, l'entité receveuse DEVRAIT se rappeler de cette valeur comme valeur par défaut pour le flux initial et le flux de réponse ; si l'attribut n'est pas inclus, l'entité receveuse DEVRAIT utiliser une valeur par défaut configurable pour les deux flux, qu'elle DOIT communiquer dans l'en-tête pour le flux de réponse. Pour toutes les strophes envoyées sur le flux initial, si l'entité initiatrice n'inclut pas d'attribut 'xml:lang', l'entité receveuse DEVRAIT appliquer la valeur par défaut ; si l'entité initiatrice inclut un attribut 'xml:lang', l'entité receveuse NE DOIT PAS la modifier ou la supprimer (voir aussi xml:lang (paragraphe 9.1.5)). La valeur de l'attribut 'xml:lang' DOIT être un NMTOKEN (comme défini au paragraphe 3.3.1 de [XML]) et DOIT se conformer au format défini dans la [RFC3066].

version la présence de l'attribut version réglé à une valeur d'au moins "1.0" signale la prise en charge de protocoles en rapport avec les flux (incluant les caractéristiques de flux) définis dans la présente spécification. Les règles détaillées concernant la génération et le traitement de cet attribut sont définies ci-dessous.

On peut les résumer comme suit :

	initiateur à receveur	**receveur à initiateur**
to	nom d'hôte du receveur	ignoré en silence
from	ignoré en silence	nom d'hôte du receveur
id	ignoré en silence	clé de session
xml:lang	langage par défaut	langage par défaut
version	signale la prise charge de XMPP 1.0	signale la prise charge de XMPP 1.0

4.4.1 Prise en charge de version

La version de XMPP spécifiée ici est "1.0" ; en particulier, cela encapsule les protocoles en rapport avec les flux (Utilisation de TLS (Section 5), Utilisation de SASL (Section 6), et Erreurs de flux (paragraphe 4.7)) ainsi que la sémantique des trois types de strophe XML définis (<message/>, <presence/>, et <iq/>). Le schéma de numérotation des versions XMPP est "<majeur>.<mineur>". Les numéros majeur et mineur DOIVENT être traités comme des entiers séparés et chaque numéro PEUT être incrémenté de plus d'un seul chiffre. Donc, "XMPP 2.4" serait une version inférieure à "XMPP 2.13", qui à son tour serait inférieure à "XMPP 12.3". Les zéros en tête (par exemple, "XMPP 6.01") DOIVENT être ignorés par les receveurs et NE DOIVENT PAS être envoyés.

Le numéro de version majeur devrait être incrémenté seulement si le flux et les formats de strophe ou les actions requises ont changé de façon si importante qu'une entité de version plus ancienne ne serait pas capable d'interopérer avec une entité de la version plus récente si elle ignorait simplement les éléments et attributs qu'elle ne comprend pas et effectuait les actions spécifiées dans l'ancienne spécification. Le numéro de version mineur indique de nouvelles capacités, et DOIT être ignoré par une entité qui a un numéro de version mineure inférieur, mais être utilisé pour des besoins d'information par l'entité qui a le plus grand numéro de version mineure. Par exemple, un numéro de version mineure peut indiquer la capacité de traiter une valeur nouvelle définie pour l'attribut de 'type' pour les strophes message, presence, ou IQ ; l'entité avec le plus grand numéro

de version mineur va simplement noter que son correspondant ne sera pas capable de comprendre cette valeur de l'attribut 'type' et ne va donc pas l'envoyer.

Les règles suivantes s'appliquent à la génération et au traitement de l'attribut 'version' au sein des en-têtes de flux par les mises en œuvre :

1. L'entité initiatrice DOIT régler la valeur de l'attribut 'version' sur l'en-tête du flux initial au plus haut numéro de version qu'il accepte (par exemple, si le plus fort numéro de version qu'il prend en charge est celui défini dans la présente spécification, il DOIT régler la valeur à "1.0").
2. L'entité receveuse DOIT régler la valeur de l'attribut 'version' sur l'en-tête du flux de réponse soit à la valeur fournie par l'entité initiatrice, soit au plus fort numéro de version pris en charge par l'entité receveuse, quel que soit le plus faible. L'entité receveuse DOIT effectuer une comparaison numérique sur les numéros de version majeure et mineure, et non une confrontation de chaîne de "<majeur>.<mineur>".
3. Si le numéro de version inclus dans l'en-tête de flux de réponse est inférieur d'au moins d'une version majeure au numéro de version inclus dans l'en-tête de flux initial et si les entités de version plus récente ne peuvent pas interopérer avec les entités de version plus ancienne comme décrit ci-dessus, l'entité initiatrice DEVRAIT générer une erreur de flux <version non prise en charge/> et terminer le flux XML et les connexions TCP sous-jacentes.
4. Si l'une ou l'autre entité reçoit un en-tête de flux sans attribut 'version', l'entité DOIT considérer la version prise en charge par l'autre entité comme étant "0.0" et NE DEVRAIT PAS inclure d'attribut 'version' dans l'en-tête de flux qu'elle envoie en réponse.

4.5 Déclarations d'espace de noms

L'élément de flux DOIT posséder à la fois une déclaration d'espace de noms de flux et une déclaration d'espace de noms par défaut ("déclaration d'espace de noms" est défini dans la spécification d'espace de noms XML [XML-NAMES]). Pour des informations détaillées sur les espaces de noms et espaces de noms par défaut de flux, voir le paragraphe 11.2 "Noms et préfixes d'espace de noms").

4.6 Caractéristiques de flux

Si l'entité initiatrice inclut l'attribut 'version' réglé à une valeur d'au moins "1.0" dans l'en-tête de flux initial, l'entité receveuse DOIT envoyer un élément fils <features/> (préfixé par le préfixe d'espace de noms du flux) à l'entité initiatrice afin d'annoncer toutes les caractéristiques de niveau flux qui peuvent être négociées (ou les capacités qui doivent par ailleurs être annoncées). Actuellement, ceci n'est utilisé que pour annoncer l'utilisation de TLS (Section 5), l'utilisation de SASL (Section 6), et le lien de ressource (Section 7) comme défini ici, et pour l'établissement de session comme défini dans la [RFC3921] ; cependant, la fonction de caractéristiques de flux pourrait être utilisée pour annoncer d'autres caractéristiques négociables à l'avenir. Si une entité ne comprend pas ou ne prend pas en charge certaines caractéristiques, elle DEVRAIT les ignorer en silence. Si une ou plusieurs caractéristiques de sécurité (par exemple, TLS et SASL) doivent réussir à être négociées avant qu'une caractéristique qui n'est pas en rapport avec la sécurité (par exemple, un lien de ressource) puisse être offerte, la caractéristique qui n'est pas en rapport avec la sécurité NE DEVRAIT PAS être incluse dans les caractéristiques de flux qui sont annoncées avant que les caractéristique de sécurité pertinentes aient été négociées.

4.7 Erreurs de flux

L'élément de flux racine PEUT contenir un élément fils <error/> qui a comme préfixe le préfixe de l'espace de noms du flux. L'erreur fille DOIT être envoyée par une entité conforme (généralement un serveur plutôt qu'un client) si elle estime qu'une erreur de niveau flux s'est produite.

4.7.1 Règles

Les règles suivantes s'appliquent aux erreurs de niveau flux :

- On suppose que toutes les erreurs de niveau flux sont irrécupérables ; donc, si une erreur se produit au niveau du flux, l'entité qui détecte l'erreur DOIT envoyer une erreur de flux à l'autre entité, envoyer une étiquette de fermeture </stream>, et terminer la connexion TCP sous-jacente.

- Si l'erreur se produit alors que le flux est en cours d'établissement, l'entité receveuse DOIT quand même envoyer l'étiquette d'ouverture <stream>, inclure l'élément <error/> comme fille de l'élément de flux, envoyer l'étiquette de fermeture </stream>, et terminer la connexion TCP sous-jacente. Dans ce cas, si l'entité initiatrice fournit un hôte inconnu dans l'attribut 'to' (ou ne fournit pas du tout d'attribut 'to') le serveur DEVRAIT fournir le nom d'hôte d'autorité du serveur dans l'attribut 'from' de l'en-tête de flux envoyé avant la terminaison.

4.7.2 Syntaxe

La syntaxe des erreurs de flux est la suivante :

```
<stream:error>
  <defined-condition xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
  <text xmlns='urn:ietf:params:xml:ns:xmpp-streams'
        xml:lang='langcode'>
    Texte descriptif FACULTATIF
  </text>
  [Élément de condition spécifique de l'application FACULTATIF]
</stream:error>
```

L'élément <error/> :

- DOIT contenir un élément fils correspondant à une des conditions d'erreur de strophe définies qui sont décrites ci-dessous ; cet élément DOIT être qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-streams' ;
- PEUT contenir un élément fils <text/> contenant des données de caractères XML qui décrivent l'erreur plus en détails ; cet élément DOIT être qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-streams' et DEVRAIT posséder un attribut 'xml:lang' qui spécifie le langage naturel des données de caractères XML ;
- PEUT contenir un élément fils pour une condition d'erreur spécifique de l'application ; cet élément DOIT être qualifié par un espace de noms défini par l'application, et sa structure est définie par cet espace de noms.

L'élément <text/> est FACULTATIF. Si il est inclus, il DEVRAIT n'être utilisé que pour fournir des informations descriptives ou de diagnostic qui complètent la signification d'une condition définie ou d'une condition spécifique de l'application. Il NE DEVRAIT PAS être interprété de façon programmatique par une application. Il NE DEVRAIT PAS être utilisé comme message d'erreur présenté à un utilisateur, mais PEUT être montré en plus du message d'erreur associé à l'élément de condition inclus.

4.7.3 Conditions définies

Les conditions d'erreur de niveau flux suivantes sont définies :

<mauvais-format/> : l'entité a envoyé un XML qui ne peut pas être traité ; cette erreur PEUT être utilisée à la place des erreurs plus spécifiques en rapport avec XML, comme <mauvais préfixe d'espace de noms/>, <xml invalide/>, <xml interdit/>, <codage non pris en charge/>, et <xml mal formé/>, bien que les erreurs plus spécifiques soient préférées.

<mauvais préfixe d'espace de noms/> : l'entité a envoyé un préfixe d'espace de noms qui n'est pas accepté, ou n'a pas envoyé de préfixe d'espace de noms sur un élément qui exige un tel préfixe (voir le paragraphe 11.2 "Noms et préfixes d'espace de noms XML").

<conflit/> : le serveur est en train de clore le flux actif pour cette entité parce qu'un nouveau flux a été initié qui est en conflit avec le flux existant.

<fin de temporisation de connexion/> : l'entité n'a pas généré de trafic sur le flux depuis un certain temps (configurable selon la politique locale de service).

<hôte parti/> : la valeur de l'attribut 'to' fourni par l'entité initiatrice dans l'en-tête de flux correspond à un nom d'hôte qui n'est plus hébergé par le serveur.

<hôte inconnu/> : la valeur de l'attribut 'to' fourni par l'entité initiatrice dans l'en-tête de flux ne correspond pas à un nom d'hôte hébergé par le serveur.

<adressage impropre/> : une strophe envoyée entre deux serveurs manque d'un attribut 'to' ou 'from' (ou l'attribut n'a pas de valeur).

<erreur interne du serveur/> : le serveur a rencontré un défaut de configuration ou une autre erreur interne indéfinie qui l'empêche de servir le flux.

<from invalide/> : le JID ou le nom d'hôte fourni dans une adresse 'from' ne correspond pas à un JID autorisé ou à un domaine validé négocié entre serveurs via SASL ou rappel, ou entre un client et un serveur via l'authentification et le lien de ressource.

<id invalide/> : l'ID de flux ou l'ID de rappel est invalide ou ne correspond pas à un ID fourni précédemment.

<espace de noms invalide/> : le nom de l'espace de noms du flux est autre que "http://etherx.jabber.org/streams" ou le nom de l'espace de noms de rappel est autre que "jabber:server:dialback" (voir le paragraphe 11.2 "Noms et préfixes d'espace de noms XML").

<invalid-xml/> : l'entité a envoyé un XML invalide sur le flux à un serveur qui effectue la validation (voir le paragraphe 11.3 "Validation").

<non autorisé/> : l'entité a tenté d'envoyer des données avant que le flux ait été authentifié, ou n'est par ailleurs pas autorisée à effectuer une action relative à la négociation de flux ; l'entité receveuse NE DOIT PAS traiter la strophe en cause avant d'envoyer l'erreur de flux.

<violation de politique/> : l'entité a violé une politique locale de service ; le serveur PEUT choisir de spécifier la politique dans l'élément <text/> ou un élément de condition spécifique de l'application.

<échec de la connexion distante/> : le serveur est incapable de se connecter correctement à une entité distante qui est exigée pour l'authentification ou l'autorisation.

<contrainte de ressource/> : il manque au serveur les ressources système nécessaires pour servir le flux.

<xml interdit/> : l'entité a tenté d'envoyer des caractéristiques XML interdites comme un commentaire, une instruction de traitement, une DTD, une référence d'entité, ou un caractère non échappé (voir le paragraphe 11.1, "Restrictions").

<voir autre hôte/> : le serveur ne va pas fournir de service à l'entité initiatrice mais redirige le trafic sur un autre hôte ; le serveur DEVRAIT spécifier le nom d'hôte ou l'adresse IP de remplacement (qui DOIT être un identifiant de domaine valide) comme données de caractères XML de l'élément <voir autre hôte/>.

<fermeture système/> : le serveur est en cours de fermeture et tous les flux actifs seront fermés.

<condition indéfinie/> : la condition d'erreur n'est pas une de celles définies par les autres conditions de cette liste ; cette condition d'erreur DEVRAIT n'être utilisée qu'en conjonction avec une condition spécifique de l'application.

<codage non accepté/> : l'entité initiatrice a codé le flux dans un codage qui n'est pas accepté par le serveur (voir le paragraphe 11.5, "Codage de caractères").

<type de strophe non accepté/> : l'entité initiatrice a envoyé un enfant de premier niveau du flux qui n'est pas accepté par le serveur.

<version non prise en charge /> : la valeur de l'attribut 'version' fourni par l'entité initiatrice dans l'en-tête de flux spécifie une version de XMPP qui n'est pas acceptée par le serveur ; le serveur PEUT spécifier la ou les versions qu'il prend en charge dans l'élément <text/>.

<xml mal formé/> : l'entité initiatrice a envoyé un XML qui n'est pas bien formé comme défini par [XML].

4.7.4 Conditions spécifiques de l'application

Comme on l'a noté, une application PEUT fournir des informations d'erreur de flux spécifiques de l'application en incluant un enfant muni d'un espace de nom approprié dans l'élément erreur. L'élément spécifique de l'application DEVRAIT complétée ou mieux qualifier un élément défini. Donc l'élément <error/> va contenir deux ou trois éléments fils :

```
<stream:error>
  <xml mal formé
     xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
  <text xml:lang='fr' xmlns='urn:ietf:params:xml:ns:xmpp-streams'>
Informations de diagnostic particulières à l'application !
  </text>
  <escape-your-data xmlns='application-ns'/>
</stream:error>
</stream:stream>
```

4.8 Exemples simplifiés de flux

Cette section contient deux exemples simplifiés d'une "session" fondée sur un flux d'un client sur un serveur (où les lignes "C" sont envoyées du client au serveur, et les lignes "S" sont envoyées du serveur au client) ; ces exemples sont inclus dans le but d'illustrer les concepts introduits jusqu'ici.

Une "session" de base :

```
C: <?xml version='1.0'?>
  <stream:stream
      to='example.com'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'
      version='1.0'>
S: <?xml version='1.0'?>
  <stream:stream
      from='example.com'
      id='someid'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'
      version='1.0'>
... chiffrement, authentification, et lien de ressource ...
C:  <message from='juliet@example.com'
            to='romeo@example.net'
            xml:lang='fr'>
C:    <body>N'est tu pas Romeo, et un Montaigue ?</body>
C:  </message>
S:  <message from='romeo@example.net'
            to='juliet@example.com'
            xml:lang='en'>
S:    <body>Ni l'un ni l'autre, beau saint, si l'un ou l'autre te déplait.</body>
S:  </message>
C: </stream:stream>
S: </stream:stream>
```

Une"session" qui tourne mal :

```
C: <?xml version='1.0'?>
  <stream:stream
      to='example.com'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'
      version='1.0'>
S: <?xml version='1.0'?>
  <stream:stream
      from='example.com'
      id='someid'
      xmlns='jabber:client'
      xmlns:stream='http://etherx.jabber.org/streams'
      version='1.0'>
... chiffrement, authentification, et lien de ressource ...
C: <message xml:lang='fr'>
    <body>Mauvais XML, pas d'étiquette de fermeture du corps !
  </message>
 S: <stream:error>
   <xml mal formé
      xmlns='urn:ietf:params:xml:ns:xmpp-streams'/>
  </stream:error>
S: </stream:stream>
```

5. Utilisation de TLS

5.1 Généralités

XMPP inclut une méthode pour sécuriser le flux contre l'altération et l'espionnage. Cette méthode de chiffrement du canal fait usage du protocole de sécurité de la couche transport (TLS, *Transport Layer Security*) [RFC2246], ainsi que d'une extension "STARTTLS" qui est modélisée d'après des extensions similaires pour les protocoles IMAP [RFC3501], POP3 [RFC1939], et ACAP [RFC2244] comme décrit dans la [RFC2595]. Le nom d'espace de noms pour l'extension STARTTLS est 'urn:ietf:params:xml:ns:xmpp-tls'.

Un administrateur d'un certain domaine PEUT exiger l'utilisation de TLS pour les communications de client à serveur, les communications de serveur à serveur, ou les deux. Les clients DEVRAIENT utiliser TLS pour sécuriser les flux avant de tenter d'achever la négociation SASL (Section 6), et les serveurs DEVRAIENT utiliser TLS entre deux domaines afin de sécuriser les communications de serveur à serveur.

Les règles suivantes s'appliquent :

1. Une entité initiatrice qui se conforme à la présente spécification DOIT inclure l'attribut 'version' réglé à la valeur "1.0" dans l'en-tête de flux initial.

2. Si la négociation TLS survient entre deux serveurs, les communications NE DOIVENT PAS se faire tant que les noms d'hôtes du système des noms de domaines (DNS, *Domain Name System*) affirmés par les serveurs n'ont pas été résolus (voir le paragraphe 14.4, "Communications de serveur à serveur").

3. Lorsque une entité receveuse qui se conforme à la présente spécification reçoit un en-tête de flux initial qui comporte l'attribut 'version' réglé à une valeur d'au moins "1.0", après l'envoi d'un en-tête de flux en réponse (incluant le fanion 'version') elle DOIT inclure un élément <starttls/> (qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-tls') ainsi que la liste des autres caractéristiques de flux qu'elle supporte.

4. Si l'entité initiatrice choisit d'utiliser TLS, la négociation TLS DOIT être achevée avant de procéder à la négociation SASL ; cet ordre de négociation est exigé pour faciliter la sauvegarde des informations d'authentification envoyées durant la négociation SASL, ainsi que pour rendre possible de fonder l'utilisation du mécanisme SASL EXTERNAL sur un certificat produit durant la négociation TLS préalable.

5. Durant la négociation TLS, une entité NE DOIT PAS envoyer de caractère espace (correspondant à la production [3] de contenu [XML]) au sein de l'élément de flux racine comme séparateurs entre les éléments (tout caractère espace montré dans les exemple TLS ci-dessous n'est inclus que pour les besoins de lisibilité) ; cette interdiction aide à s'assurer d'une précision approprié des octets de la couche de sécurité.

6. L'entité receveuse DOIT considérer que la négociation TLS a commencé immédiatement après l'envoi du caractère de fermeture ">" de l'élément <proceed/>. L'entité initiatrice DOIT considérer la négociation TLS comme ayant commencé immédiatement après avoir reçu le caractère de clôture ">" de l'élément <proceed/> de l'entité receveuse.

7. L'entité initiatrice DOIT valider le certificat présenté par l'entité receveuse ; voir le paragraphe 14.2, "Validation de certificat" concernant les procédures de validation de certificat.

8. Les certificats DOIVENT être confrontés au nom d'hôte tel que fourni par l'entité initiatrice (par exemple, un utilisateur) et non le nom d'hôte tel que résolu via le DNS ; par exemple, si l'utilisateur spécifie un nom d'hôte de "exemple.com" mais qu'une recherche de SRV DNS [RFC2782] a retourné "im.exemple.com", le certificat DOIT être vérifié comme "exemple.com". Si un JID pour toute espèce d'entité XMPP (par exemple, client ou serveur) est représenté dans un certificat, il DOIT être représenté comme chaîne UTF8 au sein d'une entité otherName à l'intérieur de subjectAltName, en utilisant l'identifiant d'objet [ASN.1] "id-on-xmppAddr" spécifié au paragraphe 5.1.1 du présent document.

9. Si la négociation TLS réussit, l'entité receveuse DOIT éliminer toute information obtenue de manière non sûre de l'entité initiatrice avant la prise d'effet de TLS.

10. Si la négociation TLS réussit, l'entité initiatrice DOIT éliminer toute information obtenue de manière non sûre de l'entité receveuse avant la prise d'effet de TLS.

11. Si la négociation TLS réussit, l'entité receveuse NE DOIT PAS offrir l'extension STARTTLS à l'entité initiatrice avec les autres caractéristiques de flux qui sont offertes lors du redémarrage du flux.

12. Si la négociation TLS réussit, l'entité initiatrice DOIT continuer avec la négociation SASL.

13. Si la négociation TLS résulte en un échec, l'entité receveuse DOIT terminer le flux XML et la connexion TCP sous-jacente.

14. Voir au paragraphe 14.7 "Technologies de mise en œuvre obligatoire" les mécanismes qui DOIVENT être pris en charge.

5.1.1 Identifiant d'objet ASN.1 pour adresse XMPP

L'identifiant d'objet [ASN.1] "id-on-xmppAddr" décrit ci-dessus est défini comme suit :

```
IDENTIFIANT D'OBJET id-pkix ::= { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5)
pkix(7) }

IDENTIFIANT D'OBJET id-on ::= { id-pkix 8 }  -- autres formes de nom
IDENTIFIANT D'OBJET id-on-xmppAddr ::= { id-on 5 }

XmppAddr ::= UTF8String
```

Cet identifiant d'objet PEUT aussi être représenté dans le format d'affichage séparé par des points comme "1.3.6.1.5.5.7.8.5".

5.2 Description

Lorsque une entité initiatrice sécurise un flux avec une entité receveuse en utilisant TLS, les étapes impliquées sont les suivantes :

1. L'entité initiatrice ouvre une connexion TCP et initie le flux par l'envoi de l'en-tête d'ouverture de flux XML à l'entité receveuse, incluant l'attribut 'version' réglé à une valeur d'au moins "1.0".

2. L'entité receveuse répond en ouvrant une connexion TCP et en envoyant un en-tête de flux XML à l'entité initiatrice, incluant l'attribut 'version' réglé à une valeur d'au moins "1.0".

3. L'entité receveuse offre l'extension STARTTLS à l'entité initiatrice en l'incluant avec la liste des autres caractéristiques de flux prises en charge (si TLS est exigé pour l'interaction avec l'entité receveuse, elle DEVRAIT signaler le fait en incluant un élément <required/> comme fils de l'élément <starttls/>).

4. L'entité initiatrice produit la commande STARTTLS (c'est-à-dire, un élément <starttls/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-tls') pour informer l'entité receveuse qu'elle souhaite commencer une négociation TLS pour sécuriser le flux.

5. L'entité receveuse DOIT répondre soit par un élément <proceed/>, soit par un élément <failure/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-tls'. Si le cas "failure" se produit, l'entité receveuse DOIT terminer le flux XML et la connexion TCP sous-jacente. Si le cas "proceed" se produit, les entités DOIVENT tenter d'achever la négociation TLS sur la connexion TCP et NE DOIVENT PAS envoyer d'autres données XML tant que la négociation TLS n'est pas achevée.

6. L'entité initiatrice et l'entité receveuse tentent d'achever une négociation TLS conformément à la [RFC2246].

7. Si la négociation TLS ne réussit pas, l'entité receveuse DOIT terminer la connexion TCP. Si la négociation TLS réussit, l'entité initiatrice DOIT initier un nouveau flux en envoyant un en-tête d'ouverture de flux XML à l'entité receveuse (il n'est pas nécessaire d'envoyer d'abord une étiquette </stream> de fermeture, car l'entité receveuse et l'entité initiatrice DOIVENT considérer le flux original comme fermé dès la réussite de la négociation TLS).

8. À réception de l'en-tête du nouveau flux de l'entité initiatrice, l'entité receveuse DOIT répondre par l'envoi d'un nouvel en-tête de flux XML à l'entité initiatrice avec les caractéristiques disponibles (mais sans inclure le dispositif STARTTLS).

5.3 Exemple de client à serveur

L'exemple suivant montre le flux de données pour un client qui sécurise un flux en utilisant STARTTLS (noter que les étapes de remplacement montrées ci-dessous sont données pour illustrer le protocole dans les cas d'échec ; elles ne sont pas exhaustives et ne seraient pas nécessairement déclenchées par les données envoyées dans l'exemple).

Étape 1 : Le client initie le flux vers le serveur :

```
<stream:stream
   xmlns='jabber:client'
   xmlns:stream='http://etherx.jabber.org/streams'
   to='example.com'
   version='1.0'>
```

Étape 2 : Le serveur répond en envoyant une étiquette de flux au client :

```
<stream:stream
   xmlns='jabber:client'
   xmlns:stream='http://etherx.jabber.org/streams'
   id='c2s_123'
   from='example.com'
   version='1.0'>
```

Étape 3 : Le serveur envoie l'extension STARTTLS au client avec les mécanismes d'authentification et toutes les autres caractéristiques de flux:

```
<stream:features>
  <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
    <required/>
  </starttls>
  <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
    <mechanism>DIGEST-MD5</mechanism>
    <mechanism>PLAIN</mechanism>
  </mechanisms>
</stream:features>
```

Étape 4 : Le client envoie la commande STARTTLS au serveur :

```
<starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

Étape 5 : Le serveur informe le client qu'il est permis de continuer :

```
<proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

Étape 5 (alt) : Le serveur informe le client de l'échec de la négociation TLS et clôt le flux et la connexion TCP :

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
</stream:stream>
```

Étape 6 : Le client et le serveur tentent d'achever la négociation TLS sur la connexion TCP existante .

Étape 7 : Si la négociation TLS réussit, le client initie un nouveau flux pour le serveur :

```
<stream:stream
   xmlns='jabber:client'
   xmlns:stream='http://etherx.jabber.org/streams'
   to='example.com'
   version='1.0'>
```

Étape 7 (alt) : Si la négociation TLS échoue, le serveur clôt la connexion TCP.

Étape 8 : Le serveur répond par l'envoi d'un en-tête de flux au client avec toutes les caractéristiques de flux disponibles :

```
<stream:stream
   xmlns='jabber:client'
   xmlns:stream='http://etherx.jabber.org/streams'
   from='example.com'
   id='c2s_234'
   version='1.0'>
<stream:features>
  <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
```

```
    <mechanism>DIGEST-MD5</mechanism>
    <mechanism>PLAIN</mechanism>
    <mechanism>EXTERNAL</mechanism>
  </mechanisms>
</stream:features>
```

Étape 9 : Le client continue avec la négociation SASL (Section 6).

5.4 Exemple de serveur à serveur

L'exemple suivant montre le flux de données pour deux serveurs qui sécurisent un flux en utilisant STARTTLS (noter que les étapes alternatives montrées ci-dessous ne sont fournies que pour illustrer des cas d'échec du protocole ; elles ne sont pas exhaustives et ne seraient pas nécessairement déclenchées par les données envoyées dans l'exemple).

Étape 1 : Le serveur1 initie le flux pour le serveur2 :

```
<stream:stream
    xmlns='jabber:server'
    xmlns:stream='http://etherx.jabber.org/streams'
    to='example.com'
    version='1.0'>
```

Étape 2 : Le serveur2 répond en envoyant une étiquette de flux au serveur1 :

```
<stream:stream
    xmlns='jabber:server'
    xmlns:stream='http://etherx.jabber.org/streams'
    from='example.com'
    id='s2s_123'
    version='1.0'>
```

Étape 3 : Le serveur2 envoie l'extension STARTTLS au serveur1 avec les mécanismes d'authentification et toutes les autres caractéristiques de flux :

```
<stream:features>
  <starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'>
    <required/>
  </starttls>
  <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
    <mechanism>DIGEST-MD5</mechanism>
    <mechanism>KERBEROS_V4</mechanism>
  </mechanisms>
</stream:features>
```

Étape 4 : Le serveur1 envoie la commande STARTTLS au serveur2 :

```
<starttls xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

Étape 5 : Le serveur2 informe le serveur1 qu'il est autorisé à poursuivre :

```
<proceed xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
```

Étape 5 (alt) : Le serveur2 informe le serveur1 que la négociation TLS a échoué et clôt le flux :

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-tls'/>
</stream:stream>
```

Étape 6 : Le serveur1 et le serveur2 tentent d'achever la négociation TLS via TCP.

Étape 7 : Si la négociation TLS a réussi, le serveur1 initie un nouveau flux pour le serveur2 :

```
<stream:stream
    xmlns='jabber:server'
```

```
   xmlns:stream='http://etherx.jabber.org/streams'
   to='example.com'
   version='1.0'>
```

Étape 7 (alt) : Si la négociation TLS a échoué, le serveur2 clôt la connexion TCP.

Étape 8 : Le serveur2 répond par l'envoi d'un en-tête de flux au serveur1 avec toutes les caractéristiques de flux disponibles :

```
<stream:stream
   xmlns='jabber:server'
   xmlns:stream='http://etherx.jabber.org/streams'
   from='example.com'
   id='s2s_234'
   version='1.0'>
<stream:features>
  <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
    <mechanism>DIGEST-MD5</mechanism>
    <mechanism>KERBEROS_V4</mechanism>
    <mechanism>EXTERNAL</mechanism>
  </mechanisms>
</stream:features>
```

Étape 9 : Le serveur1 continue par la négociation SASL (Section 6).

6. Utilisation de SASL

6.1 Généralités

XMPP inclut une méthode d'authentification d'un flux au moyen d'un profil spécifique de XMPP du protocole d'authentification simple et de couche de sécurité (SASL, *Simple Authentication et Security Layer*) [RFC2222]. SASL fournit une méthode généralisée pour ajouter la prise en charge de l'authentification aux protocoles fondés sur la connexion, et XMPP utilise un profil d'espace de noms XML générique pour SASL qui se conforme aux exigences de profilage de la [RFC2222].

Les règles suivantes s'appliquent :

1. Si la négociation SASL se fait entre deux serveurs, les communications NE DOIVENT PAS se faire tant que les noms d'hôte du système des noms de domaines (DNS, *Domain Name System*) affirmés par les serveurs n'ont pas été résolus (voir le paragraphe 14.1 "Communications de serveur à serveur").

2. Si l'entité initiatrice est capable de négociation SASL, elle DOIT inclure l'attribut 'version' réglé à une valeur d'au moins "1.0" dans l'en-tête de flux initial.

3. Si l'entité receveuse est capable de négociation SASL, elle DOIT annoncer un ou plusieurs mécanismes d'authentification au sein d'un élément <mechanisms/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-sasl' en réponse à l'étiquette de flux d'ouverture reçue de l'entité initiatrice (si l'étiquette de flux d'ouverture incluait l'attribut 'version' réglé à une valeur d'au moins "1.0").

4. Durant la négociation SASL, une entité NE DOIT PAS envoyer de caractère espace (qui correspondent au contenu [3] de la production [XML]) au sein de l'élément de flux racine comme séparateur entre éléments (tout caractère espace montré dans les exemples SASL ci-dessous n'est inclus que pour améliorer la lisibilité) ; cette interdiction aide à assurer la précision adéquate des octets de la couche de sécurité.

5. Toutes les données de caractères XML contenues dans les éléments XML utilisés durant la négociation SASL DOIVENT être codées en utilisant base64, où le codage adhère à la définition de la Section 3 de la [RFC3548].

6. Si la fourniture d'un "simple nom d'utilisateur" est acceptée par le mécanisme SASL choisi (par exemple, c'est accepté par les mécanismes DIGEST-MD5 et CRAM-MD5 mais pas par les mécanismes EXTERNAL et GSSAPI) durant l'authentification, l'entité initiatrice DEVRAIENT fourni comme simple nom d'utilisateur son domaine d'envoi (adresse IP ou nom de domaine pleinement qualifié tel que contenu dans un identifiant de domaine) dans le cas de communications de serveur à serveur, ou son nom de compte enregistré (nom d'utilisateur ou de nœud comme contenu dans un identifiant de nœud XMPP) dans le cas de communications de client à serveur.

7. Si l'entité initiatrice souhaite agir au nom d'une autre entité et si le mécanisme SASL choisi accepte la transmission d'une identité d'autorisation, l'entité initiatrice DOIT fournir une identité d'autorisation durant la négociation SASL. Si l'entité initiatrice ne souhaite pas agir au nom d'une autre entité, elle NE DOIT PAS fournir d'identité d'autorisation. Comme spécifié dans la [RFC2222], l'entité initiatrice NE DOIT PAS fournir d'identité d'autorisation sauf si l'identité d'autorisation est différente de l'identité d'autorisation par défaut déduite de l'identité d'authentification décrite dans la [RFC2222]. Si elle est fournie, la valeur de l'identité d'autorisation DOIT être de la forme <domaine> (c'est-à-dire, seulement un identifiant de domaine) pour les serveurs et de la forme <nœud@domaine> (c'est-à-dire, identifiant de nœud et identifiant de domaine) pour les clients.

8. Lors du succès d'une négociation SASL qui implique la négociation d'une couche de sécurité, l'entité receveuse DOIT éliminer toute information obtenue de l'entité initiatrice qui n'a pas été obtenue de la négociation SASL elle-même.

9. Lors du succès d'une négociation SASL qui implique la négociation d'une couche de sécurité, l'entité initiatrice DOIT éliminer toute information obtenue de l'entité receveuse qui n'a pas été obtenue de la négociation SASL elle-même.

10. Voir au paragraphe 14.4 "Technologies de mise en œuvre obligatoire" les mécanismes qui DOIVENT être pris en charge.

6.2 Description

Lorsque une entité initiatrice s'authentifie auprès d'une entité receveuse en utilisant SASL, les étapes à suivre sont :

1. L'entité initiatrice demande l'authentification SASL en incluant l'attribut 'version' dans l'en-tête de flux XML d'ouverture envoyé à l'entité receveuse, avec la valeur réglée à "1.0".

2. Après l'envoi d'un en-tête de flux XML en réponse, l'entité receveuse annonce une liste de mécanismes d'authentification SASL disponibles; dont chacun est un élément <mechanism/> inclus comme fils au sein d'un élément conteneur <mechanisms/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-sasl', qui à son tour est un fils d'un élément <features/> dans l'espace de noms des flux. Si l'utilisation de TLS (Section 5) doit être établie avant qu'un mécanisme d'authentification particulier puisse être utilisé, l'entité receveuse NE DOIT PAS fournir ce mécanisme dans la liste des mécanismes d'authentification SASL disponibles avant la négociation TLS. Si l'entité initiatrice présente un certificat valide durant la négociation TLS préalable, l'entité receveuse DEVRAIT offrir le mécanisme SASL EXTERNAL à l'entité initiatrice durant la négociation SASL (voir la [RFC2222]) bien que le mécanisme EXTERNAL PUISSE être offert aussi dans d'autres circonstances.

3. L'entité initiatrice choisit un mécanisme en envoyant un élément <auth/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-sasl' à l'entité receveuse et en incluant une valeur appropriée pour l'attribut 'mechanism'. Cet élément PEUT contenir des données de caractères XML (dans la terminologie SASL, la "réponse initiale") si le mécanisme le supporte ou l'exige ; si l'entité initiatrice a besoin d'envoyer une réponse initiale de longueur zéro, elle DOIT transmettre la réponse comme un seul signe égal ("="), qui indique que la réponse est présente mais ne contient pas de données.

4. Si nécessaire, l'entité receveuse met au défi l'entité initiatrice par l'envoi à l'entité initiatrice d'un élément <challenge/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-sasl' ; cet élément PEUT contenir des données de caractères XML (qui DOIVENT être calculées en accord avec la définition du mécanisme SASL choisi par l'entité initiatrice).

5. L'entité initiatrice répond au défi en envoyant à l'entité receveuse un élément <response/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-sasl' ; cet élément PEUT contenir des données de caractères XML (qui DOIVENT être calculées en accord avec la définition du mécanisme SASL choisi par l'entité initiatrice).

6. Si nécessaire, l'entité receveuse envoie plus de défis, et l'entité initiatrice envoie plus de réponses.

Cette série de paires de défi/réponse continue jusqu'à ce que se produisent trois choses :

1. L'entité initiatrice interrompt la prise de contact en envoyant à l'entité receveuse un élément <abort/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-sasl'. À réception d'un élément <abort/>, l'entité receveuse DEVRAIT permettre un nombre de réessais configurable mais raisonnable (au moins 2) après quoi elle DOIT mettre un terme à la connexion TCP ; cela permet à l'entité initiatrice (par exemple, un client d'utilisateur final) de tolérer des accréditifs fournis de façon incorrecte (par exemple, un mot de passe mal orthographié) sans être forcé de se reconnecter.

2. L'entité receveuse rapporte l'échec de la prise de contact en envoyant à l'entité initiatrice un élément <failure/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-sasl' (la cause particulière de l'échec DEVRAIT être communiquée dans un élément fils approprié de l'élément <failure/> comme défini au paragraphe 6.4 "Erreurs SASL"). Si le cas d'échec se produit, l'entité receveuse DEVRAIT permettre un nombre configurable mais raisonnable de réessais (au moins 2), après

quoi elle DOIT mettre un terme à la connexion TCP ; cela permet à l'entité initiatrice (par exemple, un client d'utilisateur final) de tolérer des accréditifs fournis de façon incorrecte (par exemple, un mot de passe mal orthographié) sans être forcé de se reconnecter.

3. L'entité receveuse rapporte le succès de la prise de contact en envoyant à l'entité initiatrice un élément <success/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-sasl' ; cet élément PEUT contenir des données de caractères XML (dans la terminologie SASL, des "données supplémentaires avec succès") si c'est exigé par le mécanisme SASL choisi. À réception de l'élément <success/>, l'entité initiatrice DOIT initier un nouveau flux en envoyant un en-tête de flux XML d'ouverture à l'entité receveuse (il n'est pas nécessaire d'envoyer d'abord une étiquette </stream> de clôture, car l'entité receveuse et l'entité initiatrice DOIVENT considérer que le flux d'origine est clos par l'envoi ou la réception de l'élément <success/>). À réception du nouvel en-tête de flux provenant de l'entité initiatrice, l'entité receveuse DOIT répondre par l'envoi à l'entité initiatrice d'un nouvel en-tête de flux XML, ainsi que tout dispositif disponible (mais n'incluant pas les dispositifs STARTTLS et SASL) ou un élément <features/> vide (pour signifier qu'aucun dispositif supplémentaire n'est disponible) ; tout dispositif supplémentaire de cette sorte non défini ici DOIT être défini par l'extension pertinente à XMPP.

6.3 Définition SASL

Les exigences de profilage de la [RFC2222] exigent que les informations suivantes soient fournies dans une définition de protocole :

nom du service : "xmpp"

séquence initiale : après que l'entité initiatrice a fourni un en-tête de flux XML d'ouverture et que l'entité receveuse a répondu ce qu'elle devait, l'entité receveuse fournit une liste de méthodes d'authentification acceptables. L'entité initiatrice choisit une des méthodes de la liste et l'envoie à l'entité receveuse comme valeur de l'attribut 'mechanism' possédé par un élément <auth/>, incluant facultativement une réponse initiale pour éviter un aller-retour.

séquence d'échange : les défis et réponses sont portés par l'échange d'éléments <challenge/> de l'entité receveuse à l'entité initiatrice et d'éléments <response/> de l'entité initiatrice à l'entité receveuse. L'entité receveuse rapporte les échecs en envoyant un élément <failure/> et le succès en envoyant un élément <success/> ; l'entité initiatrice interrompt l'échange par l'envoi d'un élément <abort/>. À la réussite de la négociation, les deux côtés considèrent que le flux XML d'origine est clos et de nouveaux en-tête de flux sont envoyés par les deux entités.

négociation de couche de sécurité : la couche de sécurité prend effet immédiatement après l'envoi du caractère de clôture ">" de l'élément <success/> pour l'entité receveuse, et immédiatement après la réception du caractère de clôture ">" de l'élément <success/> pour l'entité initiatrice. L'ordre des couches est d'abord TCP [RFC0793], puis TLS [RFC2246], ensuite SASL [RFC2222], enfin XMPP.

utilisation de l'identité d'autorisation : l'identité d'autorisation peut être utilisée par xmpp pour noter le <nœud@domaine> qui n'est pas celui par défaut d'un client ou l'envoi du <domaine> d'un serveur.

6.4 Erreurs SASL

Les conditions d'erreur relatives à SASL suivantes sont définies :

<aborted/> : *(interrompu)* l'entité receveuse accuse réception d'un élément <abort/> envoyé par l'entité initiatrice ; envoyé en réponse à l'élément <abort/>.

<incorrect-encoding/> : *(codage incorrect)* les données fournies par l'entité initiatrice n'ont pas pu être traitées parce que le codage base64 est incorrect (par exemple, parce que le codage ne respecte pas la définition de la Section 3 de la [RFC3548]) ; envoyé en réponse à un élément <response/> ou <auth/> avec les données de réponse initiale.

<invalid-authzid/> : *(identifiant d'autorisation invalide)* le authzid fourni par l'entité initiatrice est invalide, soit parce que il est formaté de façon incorrecte, soit parce que l'entité initiatrice n'a pas les permissions pour autoriser cet ID ; envoyé en réponse à un élément <response/> ou <auth/> avec les données de réponse initiale.

<invalid-mechanism/> : *(mécanisme invalide)* l'entité initiatrice n'a pas fourni un mécanisme ou a demandé un mécanisme qui n'est pas supporté par l'entité receveuse ; envoyé en réponse à un élément <auth/>.

<mechanism-too-weak/> : *(mécanisme trop faible)* le mécanisme demandé par l'entité initiatrice est plus faible que ce que permet la politique du serveur pour cette entité initiatrice ; envoyé en réponse à un élément <response/> ou <auth/> avec les

données de réponse initiale.

<not-authorized/> : *(non autorisé)* l'authentification a échoué parce que l'entité initiatrice n'a pas fourni des accréditifs valides (cela inclut, mais ne s'y limite pas, le cas d'un nom d'utilisateur inconnu) ; envoyé en réponse à un élément <response/> ou <auth/> avec les données de réponse initiale.

<temporary-auth-failure/> : *(échec temporaire d'authentification)* l'authentification a échoué à cause d'une condition d'erreur temporaire chez l'entité receveuse ; envoyé en réponse à un élément <auth/> ou <response/>.

6.5 Exemple de client à serveur

L'exemple suivant montre le flux de données pour un client qui s'authentifie auprès d'un serveur en utilisant SASL, normalement après une négociation TLS réussie (noter que les étapes de remplacement (alt) montrées ci-dessous ne sont données que pour illustrer les cas d'échec du protocole ; ils ne sont pas exhaustifs et ne seraient pas nécessairement déclenchés par les données envoyés dans l'exemple).

Étape 1 : Le client initie le flux pour le serveur :

```
<stream:stream
   xmlns='jabber:client'
   xmlns:stream='http://etherx.jabber.org/streams'
   to='example.com'
   version='1.0'>
```

Étape 2 : Le serveur répond par une étiquette de flux envoyée au client :

```
<stream:stream
   xmlns='jabber:client'
   xmlns:stream='http://etherx.jabber.org/streams'
   id='c2s_234'
   from='example.com'
   version='1.0'>
```

Étape 3 : Le serveur informe le client des mécanismes d'authentification disponibles :

```
<stream:features>
  <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
    <mechanism>DIGEST-MD5</mechanism>
    <mechanism>PLAIN</mechanism>
  </mechanisms>
</stream:features>
```

Étape 4 : Le client choisit un mécanisme d'authentification :

```
<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
     mechanism='DIGEST-MD5'/>
```

Étape 5 : Le serveur envoie un défi codé de la [RFC3548] au client :

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9ImF1dGgiLGNoYXJzZXQ9dXRmLTgs
YWxnb3JpdGhtPW1kNS1zZXNzCg==
</challenge>
```

Le défi décodé est :

```
realm="somerealm",nonce="OA6MG9tEQGm2hh",\
qop="auth",charset=utf-8,algorithm=md5-sess
```

Étape 5 (alt) : Le serveur retourne une erreur au client :

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <incorrect-encoding/>
```

```
</failure>
</stream:stream>
```

Étape 6 : Le client envoie une réponse codée [RFC3548] au défi :

```
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
dXNlcm5hbWU9InNvbWVub2RlIixyZWFsbT0ic29tZXJlYWxtIixub25jZT0iT0E2TUc5dEVRR20yaGgiLGNub25jZT0iT0E2
TUhYaDZWcVRyUmsiLG5jPTAwMDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5jb20iLHJlc
3BvbnNlPWQzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNoYXJzZXQ9dXRmLTgK
</response>
```

La réponse décodée est :

```
username="somenode",realm="somerealm",\
nonce="OA6MG9tEQGm2hh",cnonce="OA6MHXh6VqTrRk",\
nc=00000001,qop=auth,digest-uri="xmpp/example.com",\
response=d388dad90d4bbd760a152321f2143af7,charset=utf-8
```

Étape 7 : Le serveur envoie un autre défi codé [RFC3548] au client :

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZAo=
</challenge>
```

Le défi décodé est :

```
rspauth=ea40f60335c427b5527b84dbabcdfffd
```

Étape 7 (alt) : Le serveur retourne une erreur au client :

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <temporary-auth-failure/>
</failure>
</stream:stream>
```

Étape 8 : Le client répond au défi :

```
<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

Étape 9 : Le serveur informe le client de la réussite de l'authentification :

```
<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>
```

Étape 9 (alt) : Le serveur informe le client de l'échec de l'authentification :

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <temporary-auth-failure/>
</failure>
</stream:stream>
```

Étape 10 : Le client initie un nouveau flux avec le serveur :

```
<stream:stream
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'
    to='example.com'
    version='1.0'>
```

Étape 11 : Le serveur répond par l'envoi d'un en-tête de flux au client avec toutes les caractéristique supplémentaires (ou un élément <features> vide) :

```
<stream:stream
    xmlns='jabber:client'
    xmlns:stream='http://etherx.jabber.org/streams'
    id='c2s_345'
```

```
   from='example.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>
```

6.6 Exemple de serveur à serveur

L'exemple qui suit montre le flux de données pour un serveur qui s'authentifie auprès d'un autre serveur en utilisant SASL, normalement après la réussite de la négociation TLS (noter que les étapes de remplacement montrées ci-dessous ne sont données que pour illustrer les cas d'échec du protocole ; ils ne sont pas exhaustifs et ne seraient pas nécessairement déclenchés par les données envoyés dans l'exemple).

Étape 1 : Le serveur1 initie le flux avec le serveur2 :

```
<stream:stream
   xmlns='jabber:server'
   xmlns:stream='http://etherx.jabber.org/streams'
   to='example.com'
   version='1.0'>
```

Étape 2 : Le serveur2 répond par une étiquette de flux envoyée au serveur1 :

```
<stream:stream
   xmlns='jabber:server'
   xmlns:stream='http://etherx.jabber.org/streams'
   from='example.com'
   id='s2s_234'
   version='1.0'>
```

Étape 3 : Le serveur2 informe le serveur1 des mécanismes d'authentification disponibles :

```
<stream:features>
  <mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
    <mechanism>DIGEST-MD5</mechanism>
    <mechanism>KERBEROS_V4</mechanism>
  </mechanisms>
</stream:features>
```

Étape 4 : Le serveur1 choisit un mécanisme d'authentification :

```
<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
      mechanism='DIGEST-MD5'/>
```

Étape 5 : Le serveur2 envoie un défi codé selon la [RFC3548] au serveur1 :

```
<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
cmVhbG09InNvbWVyZWFsbSIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIixxb3A9ImF1dGgiLGNoYXJzZXQ9dXRmLTgs
YWxnb3JpdGhtPW1kNS1zZXNz
</challenge>
```

Le défi décodé est :

```
realm="somerealm",nonce="OA6MG9tEQGm2hh",\
qop="auth",charset=utf-8,algorithm=md5-sess
```

Étape 5 (alt) : Le serveur2 retourne une erreur au Serveur1 :

```
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <incorrect-encoding/>
</failure>
</stream:stream>
```

Étape 6 : Le serveur1 envoi une réponse codée selon la [RFC3548] au défi :

<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
dXNlcm5hbWU9ImV4YW1wbGUub3JnIixyZWFsbT0ic29tZXJlYWxtIixub25jZT0iT0E2TUc5dEVRR20yaGgiLGNub25jZT
0iT0E2TUhYaDZWcVRyUmsiLG5jPTAwMDAwMDAxLHFvcD1hdXRoLGRpZ2VzdC11cmk9InhtcHAvZXhhbXBsZS5vcm
ciLHJlc3BvbnNlPWQzODhkYWQ5MGQ0YmJkNzYwYTE1MjMyMWYyMTQzYWY3LGNoYXJzZXQ9dXRmLTgK
</response>

La réponse décodée est :
username="example.org",realm="somerealm",\
nonce="OA6MG9tEQGm2hh",cnonce="OA6MHXh6VqTrRk",\
nc=00000001,qop=auth,digest-uri="xmpp/example.org",\
response=d388dad90d4bbd760a152321f2143af7,charset=utf-8

Étape 7 : Le serveur2 envoie un autre défi codé [RFC3548] au serveur1 :

<challenge xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZAo=
</challenge>

Lé défi codé est :
rspauth=ea40f60335c427b5527b84dbabcdfffd

Étape 7 (alt) : Le serveur2 retourne une erreur au serveur1 :
<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <invalid-authzid/>
</failure>
</stream:stream>

Étape 8 : Le serveur1 répond au défi :

<response xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

Étape 8 (alt) : Le serveur1 interrompt la négociation :

<abort xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

Étape 9 : Le serveur2 informe le serveur1 de la réussite de l'authentification :

<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

Étape 9 (alt) : Le serveur2 informe le serveur1 de l'échec de l'authentification :

<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <aborted/>
</failure>
</stream:stream>

Étape 10 : le serveur1 initie un nouveau flux avec le serveur2 :

<stream:stream
   xmlns='jabber:server'
   xmlns:stream='http://etherx.jabber.org/streams'
   to='example.com'
   version='1.0'>

Étape 11 : Le serveur2 répond en envoyant un en-tête de flux au serveur1 avec toutes les caractéristiques supplémentaires (ou un élément <features> vide) :

<stream:stream
   xmlns='jabber:client'   xmlns:stream='http://etherx.jabber.org/streams'
   from='example.com'

```
   id='s2s_345'
   version='1.0'>
<stream:features/>
```

7. Lien de ressource

Après la négociation SASL (Section 6) avec l'entité receveuse, l'entité initiatrice PEUT vouloir ou avoir besoin de lier une ressource spécifique à ce flux. En général ceci ne s'applique qu'aux clients : afin de se conformer aux règles de format d'adressage (Section 3) et de livraison de strophes (Section 10) spécifiées ici, il DOIT y avoir un identifiant de ressource associé au <nœud@domaine> du client (qui est soit généré par le serveur, soit fourni par l'application du client) ; ceci assure que l'adresse à utiliser sur ce flux est un "JID complet" de forme <nœud@domaine/ressource>.

À réception d'une indication de succès dans la négociation SASL, le client DOIT envoyer un nouvel en-tête de flux au serveur, auquel le serveur DOIT répondre par un en-tête de flux ainsi qu'une liste des caractéristiques de flux disponibles. Précisément, si le serveur exige que le client lie une ressource au flux après le succès de la négociation SASL, il DOIT inclure un élément <bind/> vide qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-bind' dans la liste des caractéristiques de flux qu'il présente au client lors de l'envoi de l'en-tête pour le flux de réponse envoyé après la réussite de la négociation SASL (mais pas avant).

Le serveur annonce la caractéristique de lien de ressource au client :

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

Étant ainsi informé de l'exigence du lien de ressource, le client DOIT lier une ressource au flux en envoyant au serveur une strophe IQ du type "set" (voir au paragraphe 9.2.3 "Sémantique IQ") contenant des données qualifiées par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-bind'.

Si le client souhaite permettre que le serveur génère l'identifiant de ressource en son nom, il envoie une strophe IQ de type "set" qui contient un élément <bind/> vide.

Le client demande au serveur de lier une ressource :

```
<iq type='set' id='bind_1'>
  <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'/>
</iq>
```

Un serveur qui prend en charge le lien de ressource DOIT être capable de générer un identifiant de ressource au nom d'un client. Un identifiant de ressource généré par le serveur DOIT être unique pour ce <nœud@domaine>.

Si le client souhaite spécifier l'identifiant de ressource, il envoie une strophe IQ de type "set" qui contient l'identifiant de ressource désiré comme données de caractères XML d'un élément <ressource/> qui est un fils de l'élément <bind/> :

Le client lie une ressource :

```
<iq type='set' id='bind_2'>
  <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
    <resource>someresource</resource>
  </bind>
</iq>
```

Une fois que le serveur a généré un identifiant de ressource pour le client ou accepté l'identifiant de ressource fourni par le client, il DOIT retourner une strophe IQ de type "result" au client, qui DOIT inclure un élément fils <jid/> qui spécifie le JID complet pour la ressource connectée comme déterminé par le serveur.

Le serveur informe le client du succès du lien de ressource :

```
<iq type='result' id='bind_2'>
  <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>    <jid>somenode@example.com/someresource</jid>
  </bind>
</iq>
```

Un serveur DEVRAIT accepter l'identifiant de ressource fourni par le client, mais PEUT l'outrepasser avec un identifiant de ressource que le serveur génère ; dans ce cas, le serveur NE DEVRAIT PAS retourner une erreur de strophe (par exemple, <forbidden/>) au client mais plutôt DEVRAIT communiquer au client l'identifiant de ressource généré dans le résultat IQ comme montré ci-dessus.

Lorsque un client fournit un identifiant de ressource, les conditions d'erreur de strophe suivantes sont possibles (voir le paragraphe 9.3 "Erreurs de strophe") :

- o l'identifiant de ressource fourni ne peut pas être traité par le serveur conformément à Resourceprep (Appendice B) ;
- o il n'est pas permis au client de lier une ressource au flux (par exemple, parce que le nœud ou l'usager a atteint la limite du nombre de ressources connectées permis) ;
- o l'identifiant de ressource fourni est déjà utilisé mais le serveur ne permet pas le lien de plusieurs ressources connectées avec le même identifiant.

On montre ci-dessous le protocole pour ces conditions d'erreur.
L'identifiant de ressource ne peut pas être traité :

```
<iq type='error' id='bind_2'>
  <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
    <resource>someresource</resource>
  </bind>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

Le client n'a pas la permission de lier une ressource :

```
<iq type='error' id='bind_2'>
  <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
    <resource>someresource</resource>
  </bind>
  <error type='cancel'>
    <not-allowed xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

L'identifiant de ressource est déjà utilisé :

```
<iq type='error' id='bind_2'>
  <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
    <resource>someresource</resource>
  </bind>
  <error type='cancel'>
    <conflict xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
  </error>
</iq>
```

Si, avant l'achèvement de l'étape de liaison de ressource, le client tente d'envoyer une strophe XML autre qu'une strophe IQ avec un fils <bind/> qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-bind', le serveur NE DOIT PAS traiter la strophe et DEVRAIT retourner une erreur de strophe <not-authorized/> au client.

8. Rappel du serveur

8.1 Généralités

Les protocoles Jabber d'où XMPP a été adapté incluent une méthode de "rappel du serveur" pour protéger contre l'usurpation de domaine, rendant ainsi plus difficile d'usurper les strophes XML. Le rappel de serveur n'est pas un mécanisme de sécurité, et résulte seulement en une vérification faible des identités de serveur (voir les caractéristiques de sécurité de cette méthode au paragraphe 14.4 "Communications de serveur à serveur"). Les domaines qui exigent une sécurité robuste DEVRAIENT utiliser TLS et SASL ; voir les détails au paragraphe 14.4. Si SASL est utilisé pour l'authentification de serveur à serveur, le rappel NE DEVRAIT PAS être utilisé car il est alors inutile. La documentation de dialback *(rappel)* est incluse principalement pour la rétro compatibilité avec les mises en œuvre et déploiements existants.

La méthode de rappel du serveur est rendue possible par l'existence du système des noms de domaines (DNS, *Domain Name System*) car un serveur peut (normalement) découvrir le serveur d'autorité pour un certain domaine. Comme le rappel dépend du DNS, les communications inter-domaines NE DOIVENT PAS se faire tant que les noms d'hôtes du DNS affirmés par les serveurs n'ont pas été résolus (voir au paragraphe 14.4 "Communications de serveur à serveur").

Le rappel de serveur est unidirectionnel, et résulte en une vérification (faible) des identités pour un flux dans une direction. Comme le rappel de serveur n'est pas un mécanisme d'authentification, l'authentification mutuelle n'est pas possible via le rappel. Donc, le rappel du serveur DOIT être complété dans chaque direction afin de permettre une communication bidirectionnelle entre deux domaines.

La méthode pour générer et vérifier les clés utilisées dans le rappel de serveur DOIT prendre en compte les noms d'hôte utilisés, l'identifiant de flux généré par le serveur receveur, et un secret connu par le réseau du serveur d'autorité. L'identifiant de flux est critique pour la sécurité dans le rappel de serveur et DOIT donc être à la fois imprévisible et non répétitif (voir dans la [RFC1750] les recommandations sur l'aléa pour les besoins de la sécurité).

Toute erreur survenant durant la négociation de rappel DOIT être considérée comme une erreur de flux, résultant en la terminaison du flux et de la connexion TCP sous-jacente. Les conditions d'erreur possibles sont spécifiées dans la description du protocole ci-dessous.

La terminologie suivante s'applique :

serveur générateur: le serveur qui tente d'établir une connexion entre deux domaines.

serveur receveur : le serveur qui essaye d'authentifier que le serveur générateur représente le domaine qu'il prétend être.

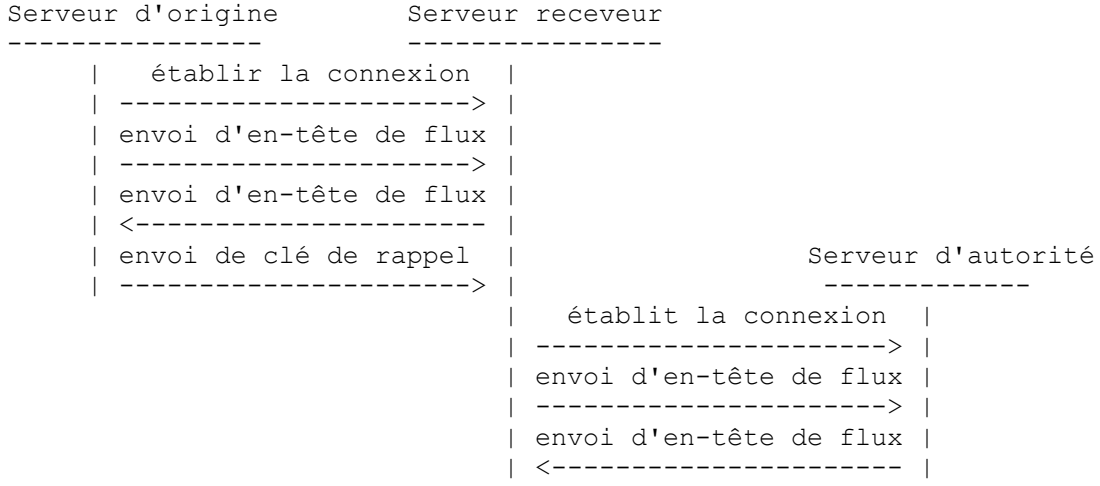
serveur d'autorité : le serveur qui répond au nom d'hôte DNS affirmé par le serveur générateur ; pour les environnements de base ce sera le serveur générateur, mais ce pourrait être une machine différente dans le réseau du serveur générateur.

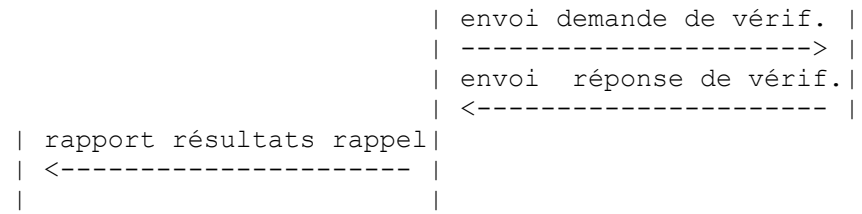
8.2 Ordre des événements

Voici un bref résumé de l'ordre des événements dans le rappel :

1. Le serveur générateur établit une connexion avec le serveur receveur.
2. Le serveur générateur envoie une valeur 'clé' sur la connexion au serveur receveur.
3. Le serveur receveur établit une connexion avec le serveur d'autorité.
4. Le serveur receveur envoie la même valeur 'clé' au serveur d'autorité.
5. Le serveur d'autorité répond que la clé est valide ou invalide.
6. Le serveur receveur informe le serveur générateur qu'il est ou non authentifié.

On peut représenter graphiquement comme suit ce flux d'événements :





8.3 Protocole

L'interaction détaillée du protocole entre les serveurs est la suivante :

1. Le serveur générateur établit une connexion TCP avec le serveur receveur.

2. Le serveur générateur envoie un en-tête de flux au serveur receveur :

```
<stream:stream
  xmlns:stream='http://etherx.jabber.org/streams'
  xmlns='jabber:server'
  xmlns:db='jabber:server:dialback'>
```

Note : Les attributs 'to' et 'from' sont FACULTATIFS sur l'élément de flux racine. L'inclusion de la déclaration d'espace de noms xmlns:db avec le nom montré indique au serveur receveur que le serveur générateur accepte le rappel. Si le nom d'espace de noms est incorrect, le serveur receveur DOIT alors générer une condition d'erreur de flux <invalid-namespace/> et terminer le flux XML et la connexion TCP sous-jacente.

3. Le serveur receveur DEVRAIT renvoyer un en-tête de flux au serveur générateur, incluant un ID univoque pour cette interaction :

```
<stream:stream
  xmlns:stream='http://etherx.jabber.org/streams'
  xmlns='jabber:server'
  xmlns:db='jabber:server:dialback'
  id='457F9224A0...'>
```

Note : Les attributs 'to' et 'from' sont FACULTATIFS sur l'élément de flux racine. Si le nom d'espace de noms est incorrect, le serveur générateur DOIT générer une condition d'erreur de flux <invalid-namespace/> et terminer le flux XML et la connexion TCP sous-jacente. Noter bien que le serveur receveur DEVRAIT répondre mais PEUT terminer en silence le flux XML et la connexion TCP sous-jacente selon les politiques de sécurité en place ; cependant, si le serveur receveur désire continuer, il DOIT renvoyer un en-tête de flux au serveur générateur.

4. Le serveur générateur envoie une clé de rappel au serveur receveur :

```
<db:result
  to='Receiving Server'
  from='serveur générateur'>
 98AF014EDC0...
</db:result>
```

Note : Cette clé n'est pas examinée par le serveur receveur, car il ne conserve pas d'information sur le serveur générateur entre les sessions. La clé générée par le serveur générateur DOIT se fonder en partie sur la valeur de l'ID fourni par le serveur receveur dans l'étape précédente, et en partie sur un secret partagé par le serveur générateur et le serveur d'autorité. Si la valeur de l'adresse 'to' ne correspond pas à un nom d'hôte reconnu par le serveur receveur, le serveur receveur DOIT alors générer une condition d'erreur de flux <host-unknown/> et terminer le flux XML et la connexion TCP sous-jacente. Si la valeur de l'adresse 'from' correspond à un domaine avec lequel le serveur receveur a déjà une connexion établie, le serveur receveur DOIT alors conserver la connexion existante jusqu'à ce qu'il valide que la nouvelle connexion est légitime ; de plus, le serveur receveur PEUT choisir de générer une condition d'erreur de flux <not-authorized/> pour la nouvelle connexion et terminer alors le flux XML et la connexion TCP sous-jacente se rapportant à la nouvelle demande.

5. Le serveur receveur établit une connexion TCP avec le nom de domaine affirmé par le serveur générateur, suite à quoi il se connecte au serveur d'autorité. (Noter qu'à titre d'optimisation, une mise en œuvre PEUT réutiliser ici une connexion

existante.)

6. Le serveur receveur envoie au serveur d'autorité un en-tête de flux :

```
<stream:stream
  xmlns:stream='http://etherx.jabber.org/streams'
  xmlns='jabber:server'
  xmlns:db='jabber:server:dialback'>
```

Note : Les attributs 'to' et 'from' sont FACULTATIFS sur l'élément de flux racine. Si le nom d'espace de noms est incorrect, le serveur d'autorité DOIT alors générer une condition d'erreur de flux <invalid-namespace/> et terminer le flux XML et la connexion TCP sous-jacente.

7. Le serveur d'autorité envoie au serveur receveur un en-tête de flux :

```
<stream:stream
  xmlns:stream='http://etherx.jabber.org/streams'
  xmlns='jabber:server'
  xmlns:db='jabber:server:dialback'
  id='1251A342B...'>
```

Note : Si le nom d'espace de noms est incorrect, le serveur receveur DOIT alors générer une condition d'erreur de flux <invalid-namespace/> et terminer le flux XML et la connexion TCP sous-jacente entre lui et le serveur d'autorité. Si une erreur de flux survient entre le serveur receveur et le serveur d'autorité, le serveur receveur DOIT alors générer une condition d'erreur de flux <remote-connection-failed/> et terminer le flux XML et la connexion TCP sous-jacente entre lui et le serveur générateur.

8 Le serveur receveur envoie au serveur d'autorité une demande de vérification d'une clé :

```
<db:verify
  from='serveur receveur'
  to='serveur générateur'
  id='457F9224A0...'>
 98AF014EDC0...
</db:verify>
```

Note : On passe ici les noms d'hôte, l'identifiant original de l'en-tête de flux du serveur receveur au serveur générateur dans l'étape 3, et la clé que le serveur générateur a envoyé au serveur receveur dans l'étape 4. Sur la base de ces informations, ainsi que des informations de secret partagé au sein du réseau du serveur d'autorité, la clé est vérifiée. Toute méthode vérifiable PEUT être utilisée pour générer la clé. Si la valeur de l'adresse 'to' ne correspond pas à un nom d'hôte reconnu par le serveur d'autorité, le serveur d'autorité DOIT alors générer une condition d'erreur de flux <host-unknown/> et terminer le flux XML et la connexion TCP sous-jacente. Si la valeur de l'adresse 'from' ne correspond pas au nom d'hôte représenté par le serveur receveur lors de l'ouverture de la connexion TCP (ou tout domaine qui en est validé, comme un sous domaine validé du nom d'hôte du serveur receveur ou un autre domaine validé hébergé par le serveur receveur) le serveur d'autorité DOIT alors générer une condition d'erreur de flux <invalid-from/> et terminer le flux XML et la connexion TCP sous-jacente.

9. Le serveur d'autorité vérifie si la clé était valide ou invalide :

```
<db:verify
  from='serveur générateur'    to='serveur receveur'
  type='valid'
  id='457F9224A0...'/>
```

ou

```
<db:verify
  from='serveur générateur'
  to='serveur receveur'
  type='invalid'
  id='457F9224A0...'/>
```

Note : Si l'ID ne correspond pas à celui fourni par le serveur receveur à l'étape 3, le serveur receveur DOIT alors générer une

condition d'erreur de flux <invalid-id/> et terminer le flux XML et la connexion TCP sous-jacente. Si la valeur de l'adresse 'to' ne correspond pas à un nom d'hôte reconnu par le serveur receveur, celui-ci DOIT générer une condition d'erreur de flux <host-unknown/> et terminer le flux XML et la connexion TCP sous-jacente. Si la valeur de l'adresse 'from' ne correspond pas au nom d'hôte représenté par le serveur générateur lors de l'ouverture de la connexion TCP (ou tout domaine qui en est validé, comme un sous domaine validé du nom d'hôte du serveur générateur ou un autre domaine validé hébergé par le serveur générateur) le serveur receveur DOIT alors générer une condition d'erreur de flux <invalid-from/> et terminer le flux XML et la connexion TCP sous-jacente. Après avoir retourné la vérification au serveur receveur, le serveur d'autorité DEVRAIT terminer le flux entre eux.

10. Le serveur receveur informe le serveur générateur du résultat :

```
<db:result
  from='serveur receveur'
  to='serveur générateur'
  type='valid'/>
```

Note : À ce point, la connexion a soit été validée via un type='valid', soit rapportée comme invalide. Si la connexion est invalide, le serveur receveur DOIT alors terminer le flux XML et la connexion TCP sous-jacente. Si la connexion est validée, les données peuvent être envoyées par le serveur générateur et lues par le serveur receveur ; avant cela, toutes les strophes XML envoyées au serveur receveur DEVRAIENT être éliminées en silence.

Le résultat de ce qui précède est que le serveur receveur a vérifié l'identité du serveur générateur, de sorte que le serveur générateur peut envoyer, et le serveur receveur peut accepter, les strophes XML sur le "flux initial" (c'est-à-dire, le flux du serveur générateur au serveur receveur). Afin de vérifier les identités des entités qui utilisent le "flux de réponse" (c'est-à-dire, le flux du serveur receveur au serveur générateur) le rappel DOIT être achevé aussi dans la direction opposée.

Après le succès de la négociation de rappel, le serveur receveur DEVRAIT accepter les paquets <db:result/> suivants (par exemple, les demandes de validation envoyées à un sous domaine ou autres noms d'hôtes desservis par le serveur receveur) provenant du serveur générateur sur la connexion validée existante ; cela permet le "portage" de la connexion validée originale dans une direction.

Même si la négociation de rappel est réussie, un serveur DOIT vérifier que toutes les strophes XML reçues de l'autre serveur incluent un attribut 'from' et un attribut 'to' ; si une strophe ne satisfait pas à cette contrainte, le serveur qui reçoit la strophe DOIT générer une condition d'erreur de flux <improper-addressing/> et terminer le flux XML et la connexion TCP sous-jacente. De plus, un serveur DOIT vérifier que l'attribut 'from' des strophes reçues de l'autre serveur inclut un domaine validé pour le flux ; si une strophe ne satisfait pas cette contrainte, le serveur qui reçoit la strophe DOIT générer une condition d'erreur de flux <invalid-from/> et terminer le flux XML et la connexion TCP sous-jacente. Ces deux vérifications aident à empêcher les usurpations relatives à des strophes particulières.

9. Strophes XML

Après une négociation TLS (Section 5) si on le désire, une négociation SASL (Section 6), et un lien de ressource (Section 7) si nécessaire, les strophes XML peuvent être envoyées sur les flux. Trois sortes de strophes XML sont définies pour les espaces de noms 'jabber:client' et 'jabber:server' : <message/>, <presence/>, et <iq/>. De plus, il y a cinq attributs communs pour ces sortes de strophes. Ces attributs communs, ainsi que la sémantique de base des trois sortes de strophes, sont définis ici ; des informations plus détaillées sur la syntaxe des strophes XML en relation avec les applications de messagerie instantanée et de présence figurent dans la [RFC3921].

9.1 Attributs communs

Les cinq attributs suivants sont communs aux strophes message, presence, et IQ :

9.1.1 to

L'attribut 'to' spécifie le JID du receveur prévu de la strophe.

Dans l'espace de noms 'jabber:client', une strophe DEVRAIT posséder un attribut 'to', mais une strophe envoyée d'un client à un serveur pour traitement par ce serveur (par exemple, presence envoyé au serveur pour diffusion à d'autres entités) NE DEVRAIT PAS posséder d'attribut 'to'.

Dans l'espace de noms 'jabber:server', une strophe DOIT posséder un attribut 'to' ; si un serveur reçoit une strophe qui ne satisfait pas à cette contrainte, il DOIT générer une condition d'erreur de flux <improper-addressing/> et terminer le flux XML et la connexion TCP sous-jacente avec le serveur en faute.

Si la valeur de l'attribut 'to' est invalide ou ne peut pas être contacté, l'entité qui découvre ce fait (généralement le serveur de l'envoyeur ou du receveur) DOIT retourner une erreur appropriée à l'envoyeur, réglant l'attribut 'from' de la strophe erronée à la valeur fournie dans l'attribut 'to' de la strophe en faute.

9.1.2 from

L'attribut 'from' spécifie le JID de l'envoyeur.

Lorsque un serveur reçoit une strophe XML dans le contexte d'un flux authentifié qualifié par l'espace de noms 'jabber:client', il DOIT faire une des deux actions suivantes :

1. valider que la valeur de l'attribut 'from' fournie par le client est celle d'une ressource connectée pour l'entité associée,
2. ajouter une adresse 'from' à la strophe dont la valeur est le JID nu (<nœud@domaine>) ou le JID complet (<nœud@domaine/ressource>) déterminé par le serveur pour la ressource connectée qui a généré la strophe (voir au paragraphe 3.5 "Détermination des adresses").

Si un client tente d'envoyer une strophe XML dont la valeur de l'attribut 'from' ne correspond pas à celle d'une des ressources connectées pour cette entité, le serveur DEVRAIT retourner une erreur de flux <invalid-from/> au client. Si un client tente d'envoyer une strophe XML sur un flux qui n'est pas encore authentifié, le serveur DEVRAIT retourner une erreur de flux <not-authorized/> au client. Si elles sont générées, ces deux conditions DOIVENT résulter en la clôture du flux et la terminaison de la connexion TCP sous-jacente ; ceci aide à empêcher les attaques de déni de service lancées à partir d'un client félon.

Lorsque un serveur génère une strophe à partir du serveur lui-même pour livraison à un client connecté (par exemple, dans le contexte de services de mémorisation de données fournis par le serveur au nom du client) la strophe DOIT soit (1) ne pas inclure d'attribut 'from', soit (2) inclure un attribut 'from' dont la valeur est le JID nu du compte (<nœud@domaine>) ou le JID complet du client (<nœud@domaine/ressource>). Un serveur NE DOIT PAS envoyer au client une strophe sans un attribut 'from' si la strophe n'a pas été générée par le serveur lui-même. Lorsque un client reçoit une strophe qui ne comporte pas d'attribut 'from', il DOIT supposer que la strophe est du serveur auquel le client est connecté.

Dans l'espace de noms 'jabber:server', une strophe DOIT posséder un attribut 'from' ; si un serveur reçoit une strophe qui ne satisfait pas à cette contrainte, il DOIT générer une condition d'erreur de flux <improper-addressing/>. De plus, la portion identifiant de domaine du JID contenu dans l'attribut 'from' DOIT correspondre au nom d'hôte du serveur envoyeur (ou de tout domaine validé de celui-ci, comme un sous domaine validé du nom d'hôte du serveur envoyeur ou un autre domaine validé hébergé par le serveur d'envoi) comme communiqué dans la négociation SASL ou la négociation de rappel ; si un serveur reçoit une strophe qui ne satisfait pas cette contrainte, il DOIT générer une condition d'erreur de flux <invalid-from/>. Ces deux conditions DOIVENT résulter en la clôture du flux et la terminaison de la connexion TCP sous-jacente ; cela aide à empêcher les attaques de déni de service lancées d'un serveur félon.

9.1.3 id

L'attribut 'id' facultatif PEUT être utilisé par une entité envoyeuse pour le traçage interne des strophes qu'elle envoie et reçoit (en particulier le suivi de l'interaction demande-réponse inhérente à la sémantique des strophes IQ). Il est FACULTATIF que la valeur de l'attribut 'id' soit unique au monde, dans un domaine, ou au sein d'un flux. La sémantique des strophes IQ impose des restrictions supplémentaires ; voir au paragraphe 9.2.3 "Sémantique de IQ".

9.1.4 type

L'attribut 'type' spécifie des informations détaillées sur l'objet ou le contexte du message, présence, ou strophe IQ. Les valeurs particulières admises pour l'attribut 'type' varient selon que la strophe est un message, une présence, ou une IQ ; les valeurs pour les strophes de message et de présence sont spécifiques des applications de messagerie instantanée et de présence et sont donc définies dans la [RFC3921], tandis que les valeurs des strophes IQ spécifient le rôle d'une strophe IQ dans une "conversation" structurée de demandes-réponses et sont donc définies au paragraphe 9.2.3 "Sémantique de IQ" ci-dessous. La seule valeur de 'type' commune aux trois strophes est "error" ; voir au paragraphe 9.3 "Erreurs de strophes".

9.1.5 xml:lang

Une strophe DEVRAIT posséder un attribut 'xml:lang' (comme défini au paragraphe 2.12 de [XML]) si la strophe contient des

données de caractères XML qui sont destinées à être présentées à un utilisateur humain (comme expliqué dans la [RFC2277], "l'internationalisation est pour les humains"). La valeur de l'attribut 'xml:lang' spécifie le langage par défaut de toutes données de caractères XML lisibles par l'homme, qui PEUVENT être outrepassées par l'attribut 'xml:lang' d'un élément fils spécifique. Si une strophe ne possède pas d'attribut 'xml:lang', une mise en œuvre DOIT supposer que le langage par défaut est celui spécifié pour le flux comme défini au paragraphe 4.4 "Attributs de flux" ci-dessus. La valeur de l'attribut 'xml:lang' DOIT être un NMTOKEN et DOIT se conformer au format défini dans la [RFC3066].

9.2 Sémantique de base

9.2.1 Sémantique de message

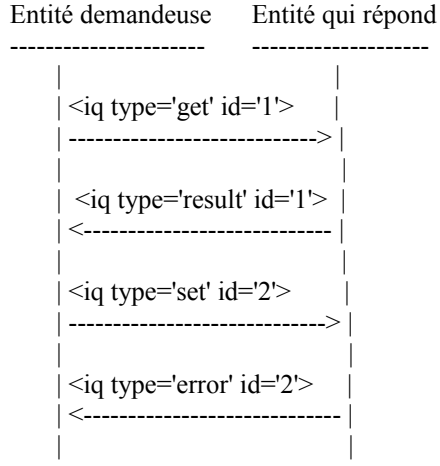
Le type de strophe <message/> peut être vu comme un mécanisme de "poussée" par lequel une entité pousse des informations vers une autre entité, d'une façon similaire aux communications qui se font dans un système comme une messagerie électronique. Toutes les strophes de message DEVRAIENT posséder un attribut 'to' qui spécifie le receveur désigné du message ; à réception d'une telle strophe, un serveur DEVRAIT l'acheminer ou la livrer au receveur désigné (voir à la section 10 "Règles de traitement des strophes XML par le serveur" les règles générales d'acheminement et de livraison des strophes XML).

9.2.2 Sémantique de présence

L'élément <presence/> peut être vu comme un mécanisme de base de diffusion ou "souscription de publication", par lequel plusieurs entités reçoivent des informations sur une entité à laquelle elles se sont abonnées (dans ce cas, des informations de disponibilité du réseau). En général, une entité qui publie DEVRAIT envoyer une strophe de présence sans attribut 'to', auquel cas le serveur auquel l'entité est connectée DEVRAIT diffuser ou multiplexer cette strophe à toutes les entités abonnées. Cependant, une entité qui publie PEUT aussi envoyer une strophe de présence avec un attribut 'to', auquel cas le serveur DEVRAIT acheminer ou livrer cette strophe au receveur désigné. Voir à la section 10 "Règles de traitement des strophes XML par le serveur" les règles générales d'acheminement et de livraison des strophes XML, et dans la [RFC3921] les règles spécifiques de présence dans le contexte d'une application de messagerie instantanée et de présence.

9.2.3 Sémantique de IQ

Information/Interrogation (IQ, *Info/Query*) est un mécanisme d'interrogation-réponse, similaire d'une certaine façon à celui de la [RFC2616]. La sémantique de IQ permet à une entité de faire une demande à une autre entité, et d'en recevoir une réponse. Les données du contenu de la demande et de la réponse sont définies par la déclaration d'espace de noms d'un élément qui est un fils direct de l'élément IQ, et l'interaction est suivi par l'entité demandeuse grâce à l'utilisation de l'attribut 'id'. Donc, les interactions d'IQ suivent un schéma commun d'échange de données structuré comme un get/result ou set/result (bien qu'une erreur puisse être retournée dans une réponse si c'est approprié) :



Pour mettre en application cette sémantique, les règles suivantes s'appliquent :

1. L'attribut 'id' est EXIGÉ pour les strophes IQ.

2. L'attribut 'type' est EXIGÉ pour les strophes IQ. Sa valeur DOIT être une des suivantes :
   * get – La strophe est une demande d'informations ou d'exigences.
   * set – La strophe donne les données requises, établit de nouvelles valeurs, ou remplace des valeurs existantes.

* result – La strophe est une réponse à une demande get ou set réussie.
* error – Une erreur s'est produite concernant le traitement ou la livraison d'un get ou set envoyé précédemment (voir au paragraphe 9.3 "Erreurs de strophe").

3. Une entité qui reçoit une demande IQ de type "get" ou "set" DOIT répondre par une réponse IQ de type "result" ou "error" (la réponse DOIT préserver l'attribut 'id' de la demande).

4. Une entité qui reçoit une strophe de type "result" ou "error" NE DOIT PAS répondre à la strophe par l'envoi d'une autre réponse IQ de type "result" ou "error" ; cependant, comme on l'a montré ci-dessus, l'entité demandeuse PEUT envoyer une autre demande (par exemple, une IQ de type "set" afin de fournir les informations requises découvertes par une paire get/result).

5. Une strophe IQ de type "get" ou "set" DOIT contenir un et seulement un élément fils qui spécifie la sémantique de la demande ou réponse particulière.

6. Une strophe IQ de type "result" DOIT inclure zéro ou un élément fils.

7. Une strophe IQ de type "error" DEVRAIT inclure l'élément fils contenu dans le "get" ou "set" associé et DOIT inclure un <error/> fils ; pour les détails, voir le paragraphe qui suit.

9.3 Erreurs de strophe

Les erreurs relatives aux strophes sont traitées d'une manière similaire à celle des erreurs de flux (paragraphe 4.7). Cependant, à la différence des erreurs de flux, les erreurs de strophes sont récupérables ; les erreurs de strophes incluent donc des conseils sur les actions que peut effectuer l'envoyeur d'origine afin de remédier à l'erreur.

9.3.1 Règles

Les règles suivantes s'appliquent aux erreurs relatives aux strophes :

o L'entité qui reçoit ou traite et détecte une condition d'erreur en relation avec une strophe DOIT retourner à l'entité envoyeuse une strophe de la même sorte (message, presence, ou IQ) dont l'attribut 'type' est réglé à une valeur de "error" (une telle strophe est appelée ici "strophe d'erreur").

o L'entité qui a généré une strophe d'erreur DEVRAIT inclure le XML envoyé d'origine de façon que l'envoyeur puisse l'inspecter et, si nécessaire, le corriger avant de tenter de l'envoyer à nouveau.

o Une strophe d'erreur DOIT contenir un élément fils <error/>.

o Un élément <error/> fils NE DOIT PAS être inclus si l'attribut 'type' a une valeur autre que "error" (ou si il n'y a pas d'attribut 'type').

o Une entité qui reçoit une strophe d'erreur NE DOIT PAS répondre à la strophe par une autre strophe d'erreur ; ceci est destiné à empêcher les boucles.

9.3.2 Syntaxe

La syntaxe des erreurs en rapport avec les strophes est la suivante :

```
<stanza-kind to='sender' type='error'>
[Il est RECOMMANDÉ d'inclure ici le XML envoyé]
  <error type='error-type'>
    <defined-condition xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'
          xml:lang='langcode'>
      texte descriptif FACULTATIF
    </text>
    [Élément de condition spécifique de l'application FACULTATIF]
  </error>
</stanza-kind>
```

Le type de strophe est soit message, soit presence, soit iq.

La valeur de l'attribut 'type' de l'élément <error/> DOIT être une des suivantes :
- cancel : ne pas réessayer (l'erreur est irrécupérable)
- continue : continuer (la condition était seulement un avertissement)
- modify : réessayer après avoir changé les données envoyées
- auth : réessayer après fourniture d'accréditifs
- wait : réessayer après attente (l'erreur est temporaire)

L'élément <error/> :

- DOIT contenir un élément fils correspondant à une des conditions d'erreur de strophe définies spécifiées ci-dessous ; cet élément DOIT être qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-stanzas'.

- PEUT contenir un élément fils <text/> contenant des données de caractères XML qui décrivent l'erreur plus en détails ; cet élément DOIT être qualifié par l'espace de noms 'urn:ietf:params:xml:ns:xmpp-stanzas' et DEVRAIT posséder un attribut 'xml:lang'.

- PEUT contenir un élément fils pour une condition d'erreur spécifique de l'application ; cet élément DOIT être qualifié par un espace de noms défini par l'application, et sa structure est définie par cet espace de noms.

L'élément <text/> est FACULTATIF. S'il est inclus, il DEVRAIT n'être utilisé que pour fournir des informations descriptives ou de diagnostic qui complètent la signification d'une condition définie ou d'une condition spécifique de l'application. Il NE DEVRAIT PAS être interprété automatiquement par une application. Il NE DEVRAIT PAS être utilisé comme message d'erreur présenté à un utilisateur, mais PEUT être montré en plus du message d'erreur associé à ou aux éléments de condition d'erreur inclus.

Finalement, pour conserver la rétro compatibilité, le schéma (spécifié dans la [RFC3921]) permet l'inclusion facultative d'un attribut 'code' sur l'élément <error/>.

9.3.3 Conditions définies

Les conditions suivantes sont définies pour l'utilisation des erreurs de strophe.

<bad-request/> : *(mauvaise demande)* l'envoyeur a envoyé un XML mal formé ou qui ne peut être traité (par exemple, une strophe IQ qui comporte une valeur non reconnue de l'attribut 'type') ; le type d'erreur associé DEVRAIT être "modify".

<conflict/> : *(conflit)* l'accès ne peut être accordé parce qu'une ressource ou session existe avec le même nom ou adresse ; le type d'erreur associé DEVRAIT être "cancel".

<feature-not-implemented/> : *(caractéristique non mise en œuvre)* la caractéristique demandée n'est pas mise en œuvre par le receveur ou serveur et ne peut donc pas être traitée ; le type d'erreur associé DEVRAIT être "cancel".

<forbidden/> : *(interdit)* l'entité demandeuse ne possède pas les permissions requises pour effectuer l'action ; le type d'erreur associé DEVRAIT être "auth".

<gone/> : *(parti)* le receveur ou serveur ne peut plus être contacté à cette adresse (la strophe d'erreur PEUT contenir une nouvelle adresse dans les données de caractères XML de l'élément <gone/>) ; le type d'erreur associé DEVRAIT être "modify".

<internal-server-error/>: *(erreur interne du serveur)* le serveur n'a pas pu traiter la strophe à cause d'une mauvaise configuration ou d'une autre erreur interne indéfinie du serveur ; le type d'erreur associé DEVRAIT être "wait".

<item-not-found/> : *(élément non trouvé)* le JID adressé ou l'élément demandé n'a pas pu être trouvé ; le type d'erreur associé DEVRAIT être "cancel".

<jid-malformed/> : *(JID mal formé)* l'entité envoyeuse a fourni ou communiqué une adresse XMPP (par exemple, une valeur de l'attribut 'to') ou un de ses aspects (par exemple, un identifiant de ressource) qui ne respecte pas la syntaxe définie dans le schéma d'adressage (Section 3) ; le type d'erreur associé DEVRAIT être "modify".

<not-acceptable/> : *(non acceptable)* le receveur ou serveur comprend la demande mais refuse de la traiter parce qu'elle ne respecte pas les critères définis par le receveur ou serveur (par exemple, une politique locale concernant les mots

acceptables dans un message) ; le type d'erreur associé DEVRAIT être "modify".

<not-allowed/> : *(non permis)* le receveur ou serveur ne permet à aucune entité d'effectuer l'action ; le type d'erreur associé DEVRAIT être "cancel".

<not-authorized/> : *(non autorisé)* l'envoyeur doit fournir des accréditifs appropriés avant d'être autorisé à effectuer l'action, ou a fourni des accréditifs impropres ; le type d'erreur associé DEVRAIT être "auth".

<payment-required/> : *(payement exigé)* l'entité demandeuse n'est pas autorisée à accéder au service demandé parce qu'un payement est exigé ; le type d'erreur associé DEVRAIT être "auth".

<recipient-unavailable/> : *(receveur indisponible)* le receveur désigné est temporairement indisponible ; le type d'erreur associé DEVRAIT être "wait" (noter qu'une application NE DOIT PAS retourner cette erreur si le faire donnerait des informations sur la disponibilité du réseau du receveur désigné à une entité qui n'est pas autorisée à connaître de telles informations).

<redirect/> : *(redirection)* le receveur ou serveur redirige les demandes sur ces informations sur une autre entité, généralement de façon temporaire (la strophe d'erreur DEVRAIT contenir l'adresse de remplacement, qui DOIT être un JID valide, dans les données de caractères XML de l'élément <redirect/>) ; le type d'erreur associé DEVRAIT être "modify".

<registration-required/> : *(enregistrement exigé)* l'entité demandeuse n'est pas autorisée à accéder au service demandé parce que l'enregistrement est exigé ; le type d'erreur associé DEVRAIT être "auth".

<remote-server-not-found/> : *(serveur distant non trouvé)* un serveur ou service distant spécifié au titre de partie ou de tout le JID du receveur désigné n'existe pas ; le type d'erreur associé DEVRAIT être "cancel".

<remote-server-timeout/> : *(fin de temporisation du serveur distant)* un serveur ou service distant spécifié au titre de partie ou de tout le JID du receveur désigné (ou exigé pour satisfaire une demande) n'a pas pu être contacté dans un délai raisonnable ; le type d'erreur associé DEVRAIT être "wait".

<resource-constraint/> : *(contraintes de ressource)* le serveur ou receveur manque des ressources système nécessaires pour servir la demande ; le type d'erreur associé DEVRAIT être "wait".

<service-unavailable/> : *(service indisponible)* le serveur ou receveur ne fournit pas actuellement le service demandé ; le type d'erreur associé DEVRAIT être "cancel".

<subscription-required/> : *(abonnement exigé)* l'entité demandeuse n'est pas autorisée à accéder au service demandé parce qu'un abonnement est exigé ; le type d'erreur associé DEVRAIT être "auth".

<undefined-condition/> : *(condition indéfinie)* la condition d'erreur n'est pas une de celles définies par les autres conditions de cette liste ; tout type d'erreur peut être associé à cette condition, et elle ne DEVRAIT être utilisée qu'en conjonction avec une condition spécifique de l'application.

<unexpected-request/> : *(demande inattendue)* le receveur ou serveur comprend la demande mais ne l'attendait pas à ce moment (par exemple, la demande est déclassée) ; le type d'erreur associé DEVRAIT être "wait".

9.3.4 Conditions spécifiques de l'application

Comme on l'a noté, une application PEUT fournir des informations d'erreur de strophe spécifiques de l'application en incluant un élément fils d'un espace de noms approprié dans l'élément d'erreur. L'élément spécifique de l'application DEVRAIT compléter ou améliorer la qualification d'un élément défini. Donc, l'élément <error/> va contenir deux ou trois éléments fils :

```
<iq type='error' id='some-id'>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <too-many-parameters xmlns='application-ns'/>
  </error>
</iq>

<message type='error' id='another-id'>
  <error type='modify'>
    <undefined-condition
        xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
```

```
   <text xml:lang='en'
         xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
     Informations de diagnostic particulières à l'application...
   </text>
   <special-application-condition xmlns='application-ns'/>
  </error>
</message>
```

10. Règles du serveur pour le traitement des strophes XML

Les mises en œuvre de serveur conformes DOIVENT s'assurer du traitement dans l'ordre des strophes XML entre deux entités.

Au delà de l'exigence du traitement dans l'ordre, chaque mise en œuvre de serveur va contenir sa propre "arborescence de livraison" pour traiter les strophes qu'elle reçoit. Une telle arborescence détermine si une strophe doit être acheminée à un autre domaine, traitée en interne, ou livrée à une ressource associée à un nœud connecté. Les règles suivantes s'appliquent :

10.1 Pas d'adresse 'to'

Si la strophe ne possède pas d'attribut 'to', le serveur DEVRAIT la traiter au nom de l'entité qui l'a envoyée. Comme toutes les strophes reçues des autres serveurs DOIVENT posséder un attribut 'to', cette règle ne s'applique qu'aux strophes reçues d'une entité enregistrée (comme un client) qui est connectée au serveur. Si le serveur reçoit une strophe de présence sans un attribut 'to', le serveur DEVRAIT la diffuser aux entités qui sont abonnées à la présence de l'entité envoyeuse, si applicable (la sémantique de la diffusion de présence pour les applications de messagerie instantanée et de présence est définie dans la [RFC3921]). Si le serveur reçoit une strophe IQ du type "get" ou "set" sans attribut 'to' et si elle comprend l'espace de noms qui qualifie le contenu de la strophe, il DOIT soit traiter la strophe au nom de l'entité envoyeuse (où la signification de "traiter" est déterminé par la sémantique de l'espace de noms qualifiant) soit retourner une erreur à l'entité envoyeuse.

10.2 Domaine étranger

Si le nom d'hôte de la portion identifiant de domaine du JID contenue dans l'attribut 'to' ne correspond pas à un des noms d'hôte configurés du serveur lui-même ou d'un de ses sous domaines, le serveur DEVRAIT acheminer la strophe au domaine étranger (sous réserve du provisionnement du service local et des politiques de sécurité concernant la communication inter domaines). Il y a deux cas possibles :

Il existe déjà un flux de serveur à serveur entre les deux domaines : le serveur de l'envoyeur achemine la strophe au serveur d'autorité pour le domaine étranger sur le flux existant.

Il n'existe pas de flux de serveur à serveur entre les deux domaines : le serveur de l'envoyeur (1) résout le nom d'hôte du domaine étranger (comme défini au paragraphe 14.4 "Communications de serveur à serveur") (2) négocie un flux de serveur à serveur entre les deux domaines (comme défini à la Section 5 "Utilisation de TLS" et à la Section 6 "Utilisation de SASL") et (3) achemine la strophe au serveur d'autorité pour le domaine étranger sur le nouveau flux établi.

En cas d'échec de l'acheminement au routeur du receveur, le serveur de l'envoyeur DOIT retourner une erreur à l'envoyeur ; si le serveur du receveur peut être contacté mais si la livraison par le serveur du receveur échoue, le serveur du receveur DOIT retourner une erreur à l'envoyeur au moyen du serveur de l'envoyeur.

10.3 Sous domaine

Si le nom d'hôte de la portion identifiant de domaine du JID contenu dans l'attribut 'to' correspond à un sous domaine d'un des noms d'hôte configuré du serveur lui-même, le serveur DOIT soit traiter la strophe elle-même soit acheminer la strophe à un service spécialisé qui est chargé de ce sous domaine (si le sous domaine est configuré) soit retourner une erreur à l'envoyeur (si le sous domaine n'est pas configuré).

10.4 Simple domaine ou ressource spécifique

Si le nom d'hôte de la portion identifiant de domaine du JID contenu dans l'attribut 'to' correspond à un nom d'hôte configuré du serveur lui- même et si le JID contenu dans l'attribut 'to' est de la forme <domaine> ou <domaine/ressource>, le serveur (ou une ressource qui en est définie) DOIT soit traiter la strophe comme approprié pour la sorte de strophe, soit retourner une strophe d'erreur à l'envoyeur.

10.5 Nœud dans le même domaine

Si le nom d'hôte de la portion identifiant de domaine du JID contenu dans l'attribut 'to' correspond à un nom d'hôte configuré du serveur lui-même et si le JID contenu dans l'attribut 'to' est de la forme <nœud@domaine> ou <nœud@domaine/ressource>, le serveur DEVRAIT livrer la strophe au receveur désigné de la strophe comme représenté par le JID contenu dans l'attribut 'to'. On applique les règles suivantes :

1. Si le JID contient un identifiant de ressource (c'est-à-dire, est de la forme <nœud@domaine/ressource>) et si il existe une ressource connectée qui correspond au JID complet, le serveur du receveur DEVRAIT livrer la strophe au flux ou session qui correspond exactement à l'identifiant de ressource.

2. Si le JID contient un identifiant de ressource et si il n'existe pas de ressource connectée qui corresponde au JID complet, le serveur du receveur DEVRAIT retourner une erreur de strophe <service-unavailable/> à l'envoyeur.

3 Si le JID est de la forme <nœud@domaine> et si il existe au moins une ressource connectée pour le nœud, le serveur du receveur DEVRAIT livrer la strophe à au moins une des ressources connectées, selon les règles spécifiques de l'application (un ensemble de règles de livraison pour les applications de messagerie instantanée et de présence est défini dans la [RFC3921]).

11. Utilisation de XML au sein de XMPP

11.1 Restrictions

XMPP est un protocole simplifié et spécialisé pour écouler les éléments XML afin d'échanger des informations structurées presque en temps réel. Parce que XMPP n'exige pas l'analyse de documents XML arbitraires et complets, il n'est pas exigé que XMPP doive prendre en charge l'ensemble complet des caractéristique de [XML]. En particulier, les restrictions suivantes s'appliquent.

Par rapport à la génération de XML, une mise en œuvre XMPP NE DOIT PAS injecter dans un flux XML un des éléments suivants :
- o des commentaires (comme défini au paragraphe 2.5 de [XML])
- o des instructions de traitement (comme défini au paragraphe 2.6 de [XML])
- o des sous ensembles internes ou externes de DTD (comme défini au paragraphe 2.8 de [XML])
- o des références à des entités internes ou externes (comme défini au paragraphe 4.2 de [XML]) à l'exception d'entités prédéfinies (comme défini au paragraphe 4.6 de [XML])
- o des données de caractère ou des valeurs d'attribut contenant des caractères non échappés qui se transposent en entités prédéfinies (comme défini au paragraphe 4.6 de [XML]) ; de tels caractères DOIVENT être échappés.

À l'égard du traitement XML, si une mise en œuvre XMPP reçoit de telles données XML interdites, elle DOIT les ignorer.

11.2 Noms et préfixes d'espace de noms XML

Les espaces de noms XML [XML-NAMES] sont utilisés dans tout XML conforme à XMPP pour créer des limites strictes de propriété des données. La fonction de base des espaces de noms est de séparer les différents vocabulaires des éléments XML qui sont structurellement mélangés. S'assurer que l'XML conforme à XMPP a une capacité d'espace de noms permet à tout XML admissible d'être structurellement mélangé avec tout élément de données au sein de XMPP. Les règles des noms et préfixes d'espace de noms XML sont définies dans les paragraphes qui suivent.

11.2.1 Espace de noms de flux

Une déclaration d'espace de noms de flux est EXIGÉE dans tous les en-têtes de flux XML. Le nom de l'espace de noms de flux DOIT être 'http://etherx.jabber.org/streams'. Les noms d'élément de l'élément <stream/> et ses fils <features/> et <error/> DOIVENT être qualifiés par le préfixe d'espace de noms de flux dans toutes les instances. Une mise en œuvre DEVRAIT générer seulement le préfixe 'stream:' pour ces éléments, et pour des raisons historiques PEUVENT accepter seulement le préfixe 'stream:'.

11.2.2 Espace de noms par défaut

Une déclaration d'espace de noms par défaut est EXIGÉE et est utilisée dans tous les flux XML afin de définir les enfants de

premier niveau admissibles de l'élément racine de flux. Cette déclaration d'espace de noms DOIT être la même pour le flux initial et le flux de réponse afin que les deux flux soient qualifiés de façon cohérente. La déclaration d'espace de noms par défaut s'applique au flux et à toutes les strophes envoyées au sein d'un flux (sauf qualification explicite par un autre espace de noms, ou par le préfixe de l'espace de noms de flux ou l'espace de noms de rappel).

Une mise en œuvre de serveur DOIT prendre en charge les deux espaces de noms par défaut suivants (pour des raisons historiques, certaines mises en œuvre PEUVENT ne prendre en charge que ces deux espaces de noms par défaut) :

jabber:client : cet espace de noms par défaut est déclaré lorsque le flux est utilisé pour communiquer entre un client et un serveur.

jabber:server : cet espace de noms par défaut est déclaré lorsque le flux est utilisé pour communiquer entre deux serveurs.

Une mise en œuvre de client DOIT prendre en charge l'espace de noms par défaut 'jabber:client', et pour des raisons historiques PEUT ne prendre en charge que cet espace de noms par défaut.

Une mise en œuvre NE DOIT PAS générer de préfixes d'espace de noms pour des éléments dans l'espace de noms par défaut si l'espace de noms par défaut est 'jabber:client' ou 'jabber:server'. Une mise en œuvre NE DEVRAIT PAS générer de préfixes d'espace de noms pour des éléments qualifiés par des espaces de noms de contenu (par opposition à de flux) autres que 'jabber:client' et 'jabber:server'.

Note : les espaces de noms 'jabber:client' et 'jabber:server' sont presque identiques mais sont utilisés dans des contextes différents (communications de client à serveur pour 'jabber:client' et communications de serveur à serveur pour 'jabber:server'). La seule différence entre les deux est que les attributs 'to' et 'from' sont FACULTATIFS sur les strophes envoyées au sein de 'jabber:client', tandis qu'elles sont EXIGÉES sur les strophes au sein de 'jabber:server'. Si une mise en œuvre conforme accepte un flux qui est qualifié par l'espace de noms 'jabber:client' ou 'jabber:server', elle DOIT prendre en charge les attributs communs (paragraphe 9.1) et la sémantique de base (paragraphe 9.2) des trois sortes de strophes centrales (message, presence, et IQ).

11.2.3 Espace de noms de rappel

Une déclaration d'espace de noms de rappel est EXIGÉE pour tous les éléments utilisés dans le rappel de serveur (Section 8). Le nom de l'espace de noms de rappel DOIT être 'jabber:server:dialback'. Tous les éléments qualifiés par cet espace de noms DOIVENT avoir un préfixe. Une mise en œuvre ne DEVRAIT générer que le préfixe 'db:' pour de tels éléments et PEUT n'accepter que le préfixe 'db:'.

11.3 Validation

Sauf comme noté au sujet des adresses 'to' et 'from' pour les strophes au sein de l'espace de noms 'jabber:server', un serveur n'est pas responsable de la validation des éléments XML transmis à un client ou un autre serveur ; une mise en œuvre PEUT choisir de ne fournir que des éléments de données validés mais ceci est FACULTATIF (mais une mise en œuvre NE DOIT PAS accepter du XML mal formé). Les clients NE DEVRAIENT PAS compter sur une capacité d'envoyer des données qui ne se conforme pas aux schémas, et DEVRAIENT ignorer tout élément ou attribut non conforme sur le flux XML entrant. La validation des flux XML et des strophes est FACULTATIVE, et les schémas sont inclus ici pour de simples besoins de description.

11.4 Inclusion d'une déclaration textuelle

Les mises en œuvre DEVRAIENT envoyer une déclaration textuelle avant d'envoyer un en-tête de flux. Les applications DOIVENT suivre les règles de [XML] concernant les circonstances dans lesquelles une déclaration textuelle est incluse.

11.5 Codage de caractères

Les mises en œuvre DOIVENT prendre en charge la transformation UTF-8 [RFC3629] des caractères du jeu de caractères universel (ISO/CEI 10646-1 [UCS2]) comme exigé par la [RFC2277]. Les mises en œuvre NE DOIVENT PAS tenter d'utiliser d'autre codage.

12. Cœur des exigences de conformité

Cette section résume les aspects spécifiques du protocole extensible de messagerie instantanée et de présence qui DOIVENT être pris en charge par les serveurs et clients afin d'être considérés comme des mises en œuvre conformes ainsi que les aspects supplémentaires du protocole qui DEVRAIENT être pris en charge. Pour les besoins de la conformité, on fait une distinction entre les protocoles centraux (qui DOIVENT être pris en charge par tout serveur ou client, sans considération de l'application spécifique) et les protocoles de messagerie instantanée (qui ne DOIVENT être pris en charge que par les applications de messagerie instantanée et de présence construites par dessus les protocoles centraux). Les exigences de conformité qui s'appliquent à tous les serveurs et clients sont spécifiées dans cette section ; les exigences de conformité pour les serveurs et clients de messagerie instantanée sont spécifiées dans la section correspondante de la [RFC3921].

12.1 Serveurs

En plus de toutes les exigences définies par rapport à la sécurité, l'usage de XML, et l'internationalisation, un serveur DOIT prendre en charge les protocoles centraux suivants afin d'être considéré conforme :

- Application de la [RFC3491], des profils Nodeprep (Appendice A), et Resourceprep (Appendice B) de la [RFC3454] aux adresses (incluant de s'assurer que les identifiants de domaine sont des noms de domaine internationalisés comme défini dans la [RFC3490])
- Les flux XML (Section 4), incluant l'utilisation de TLS (Section 5), l'utilisation de SASL (Section 6), et du lien de ressource (Section 7)
- La sémantique de base des trois sortes de strophes définies (c'est-à-dire, <message/>, <presence/>, et <iq/>) comme spécifié dans la sémantique de strophe (paragraphe 9.2)
- La génération (et, lorsque approprié, le traitement) de la syntaxe et sémantique des erreurs relatives au flux, à TLS, SASL, et aux strophes XML

De plus, un serveur PEUT prendre en charge le protocole central suivant :

- Rappel du serveur (Section 8)

12.2 Clients

Un client DOIT prendre en charge les protocoles centraux suivants afin d'être considéré conforme :

- Les flux XML (Section 4), incluant l'utilisation de TLS (Section 5), l'utilisation de SASL (Section 6), et du lien de ressource (Section 7)
- La sémantique de base des trois sortes de strophes définies (c'est-à-dire, <message/>, <presence/>, et <iq/>) comme spécifié dans la sémantique de strophes (paragraphe 9.2)
- Le traitement (et, lorsque approprié, la génération) de la syntaxe et la sémantique des erreurs relatives aux flux, à TLS, à SASL, et aux strophes XML.

De plus, un client DEVRAIT prendre en charge les protocoles centraux suivants :

- La génération des adresses auxquelles la [RFC3491], les profils Nodeprep (Appendice A), et Resourceprep (Appendice B) de la [RFC3454] peuvent être appliqués sans échec.

13. Considérations d'internationalisation

Les flux XML DOIVENT être codés en UTF-8 comme spécifié au paragraphe 11.5 "Codage de caractères. Comme spécifié au paragraphe 4.4 "Attributs de flux", un flux XML DEVRAIT inclure un attribut 'xml:lang' qui est traité comme langage par défaut pour toutes les données de caractères XML envoyées sur le flux qui est destiné à être présenté à un utilisateur humain. Comme spécifié au paragraphe 9.1.5 xml:lang, une strophe XML DEVRAIT inclure un attribut 'xml:lang' si la strophe contient des données de caractères XML qui sont destinées à être présentées à un utilisateur humain. Un serveur DEVRAIT appliquer l'attribut 'xml:lang' par défaut aux strophes qu'il achemine ou livre au nom des entités connectées, et NE DOIT PAS modifier ou supprimer les attributs 'xml:lang' des strophes qu'il reçoit des autres entités.

14. Considérations sur la sécurité

14.1 Haute sécurité

Pour les besoins des communications XMPP (de client à serveur et de serveur à serveur) le terme de "haute sécurité" se réfère à l'utilisation de technologies de sécurité qui fournissent à la fois l'authentification mutuelle et la vérification de l'intégrité ; en

particulier, lors de l'utilisation d'une authentification fondée sur le certificat pour fournir une haute sécurité, une chaîne de confiance DEVRAIT être établie hors bande, bien qu'une autorité de certificat partagée qui signe les certificats pourrait permettre que des certificats précédemment inconnus établissent la confiance dans la bande. Voir au paragraphe 14.2 les procédures de validation de certificat.

Les mises en œuvre DOIVENT prendre en charge la haute sécurité. Le provisionnent de service DEVRAIT utiliser la haute sécurité, sous réserve des politiques locales de sécurité.

14.2 Validation de certificat

Lorsque un homologue XMPP communique en toute sécurité avec un autre homologue, il DOIT valider le certificat de l'homologue. Il y a trois cas possibles :

Cas n° 1 : l'homologue contient un certificat d'entité d'extrémité qui apparaît comme certifié par une chaîne de certificats qui se termine par une ancre de confiance (comme décrit au paragraphe 6.1 de la [RFC3280]).

Cas n° 2: le certificat de l'homologue est certifié par une autorité de certificat qui n'est pas connue de l'homologue valideur.

Cas n° 3: Le certificat de l'homologue est auto signé.

Dans le cas n° 1, l'homologue valideur DOIT faire une des deux choses suivantes :

1. Vérifier le certificat de l'homologue conformément aux règles de la [RFC3280]. Le certificat DEVRAIT alors être vérifié par rapport à l'identité attendue de l'homologue suivant les règles décrites dans la [RFC2818], sauf qu'une extension subjectAltName de type "xmpp" DOIT être utilisée comme identité si elle est présente. Si une de ces vérifications échoue, les clients en mode utilisateur DOIVENT notifier l'utilisateur (les clients PEUVENT donner à l'utilisateur l'opportunité de continuer la connexion dans tous les cas) ou terminer la connexion avec une erreur de "mauvais certificat". Les clients automatisés DEVRAIENT terminer la connexion (avec une erreur de "mauvais certificat") et enregistrer l'erreur dans un journal d'audit approprié. Les clients automatisés PEUVENT fournir un réglage de configuration qui désactive cette vérification, mais DOIVENT fournir un réglage qui la permette.

2. L'homologue DEVRAIT montrer le certificat pour approbation à l'utilisateur, incluant la chaîne entière de certificats. L'homologue DOIT mettre le certificat en antémémoire (ou une représentation non falsifiable telle qu'un hachage). Dans les futures connexions, l'homologue DOIT vérifier que le même certificat a été présenté et DOIT notifier à l'utilisateur si il a été changé.

Dans les cas n° 2 et n° 3, les mises en œuvre DEVRAIENT agir comme en (2) ci-dessus.

14.3 Communications de client à serveur

Une mise en œuvre de client conforme DOIT prendre en charge TLS et SASL pour les connexions à un serveur.

Le protocole TLS pour le chiffrement des flux XML (défini à la Section 5 "Utilisation de TLS") donne un mécanisme fiable pour s'assurer de la confidentialité et de l'intégrité des données échangées entre deux entités.

Le protocole SASL pour l'authentification des flux XML (défini à la Section 6 "Utilisation de SASL") donne un mécanisme fiable pour valider qu'un client qui se connecte à un serveur est bien qui il prétend être.

Les communications de client à serveur NE DOIVENT PAS se faire tant que le nom d'hôte DNS affirmé par le serveur n'a pas été résolu. De telles résolutions DEVRAIENT d'abord tenter de résoudre le nom d'hôte en utilisant un service de "xmpp-client" et proto de "tcp" [RFC2782], résultant en enregistrements de ressource tels que "_xmpp-client._tcp.example.com." (l'utilisation de la chaîne "xmpp-client" pour l'identifiant de service est cohérente avec l'enregistrement de l'IANA). Si la recherche SRV échoue, la position de repli est une résolution normale d'enregistrement d'adresse IPv4/IPv6 pour déterminer l'adresse IP, en utilisant l'accès "xmpp-client" de 5222, enregistré auprès de l'IANA.

L'adresse IP et la méthode d'accès des clients NE DOIVENT PAS être rendues publiques par un serveur, ni les connexions autres que la connexion originale du serveur requise. Cela aide à protéger le serveur du client des attaques directes ou de l'identification par des tiers.

14.4 Communications de serveur à serveur

Une mise en œuvre de serveur conforme DOIT prendre en charge TLS et SASL pour les communications inter domaine. Pour des raisons historiques, une mise en œuvre conforme DEVRAIT aussi prendre en charge le rappel de serveur (Section 8).

Parce que le provisionnement de service est une affaire de politique, il est FACULTATIF pour un certain domaine de communiquer avec d'autres domaines, et les communications de serveur à serveur PEUVENT être désactivées par l'administrateur d'une certaine mise en œuvre. Si un certain domaine permet les communications inter domaines, il DEVRAIT permettre la haute sécurité.

Les administrateurs peuvent vouloir exiger l'utilisation de SASL pour les communications de serveur à serveur afin d'assurer l'authentification et la confidentialité (par exemple, sur le réseau privé d'une organisation). Les mises en œuvre conformes DEVRAIENT prendre en charge SASL pour cela.

Les connexions inter domaines NE DOIVENT PAS se faire tant que n'ont pas été résolus les noms d'hôtes DNS affirmés par les serveurs. De telles résolutions DOIVENT d'abord tenter de résoudre le nom d'hôte en utilisant un service de "xmpp-server" et proto de "tcp" [RFC2782], résultant en des enregistrements de ressource tels que "_xmpp-server._tcp.example.com." (l'utilisation de la chaîne "xmpp-server" pour l'identifiant de service est cohérente avec l'enregistrement de l'IANA ; on notera que l'identifiant de service "xmpp-server" supplante l'utilisation antérieure d'un identifiant de service "jabber", car l'utilisation précédente ne se conformait pas à la [RFC2782] ; les mises en œuvre qui désirent être rétro compatibles devraient continuer de rechercher ou répondre aussi à l'identifiant de service "jabber"). Si la recherche SRV échoue, le repli est une résolution normale d'adresse IPv4/IPv6 pour déterminer l'adresse IP, en utilisant l'accès "xmpp-server" de 5269, enregistré auprès de l'IANA.

Le rappel de serveur aide à protéger contre l'usurpation de domaine, rendant ainsi plus difficile d'usurper des strophes XML. Ce n'est pas un mécanisme d'authentification, de sécurisation, ou de chiffrement de flux entre serveurs comme ce qui est fait via SASL et TLS, et il résulte seulement en une vérification faible des identités de serveur. De plus, il est susceptible de subir des attaques d'empoisonnement du DNS si DNSSEC [RFC2535] est utilisé, et même si les informations du DNS sont précises, le rappel ne peut pas protéger des attaques dans lesquelles l'attaquant est capable de capturer l'adresse IP du domaine distant. Les domaines qui exigent une sécurité robuste DEVRAIENT utiliser TLS et SASL. Si SASL est utilisé pour l'authentification de serveur à serveur, le rappel NE DEVRAIT PAS être utilisé car il sera inutile.

14.5 Ordre des couches

L'ordre des couches dan lesquelles les protocoles DOIVENT être mis en piles est le suivant :
1. TCP
2. TLS
3. SASL
4. XMPP

La raison de cet ordre est que la [RFC0793] est la couche de connexion de base utilisée par tous les protocoles mis en pile par dessus TCP, la [RFC2246] est souvent fournie à la couche du système d'exploitation, la [RFC2222] est souvent fournie à la couche d'application, et XMPP est l'application elle-même.

14.6 Absence de lien de canal SASL à TLS

Le cadre SASL ne fournit pas de mécanisme pour lier l'authentification SASL à une couche de sécurité fournissant la protection de la confidentialité et de l'intégrité qui ont été négociées à une couche inférieure. Ce manque d'une "liaison de canal" empêche SASL d'être capable de vérifier que les points d'extrémité de source et de destination auxquels est liée la sécurité de la couche inférieure sont équivalents aux points d'extrémité que SASL authentifie. Si les points d'extrémité ne sont pas identiques, la sécurité de la couche inférieure ne peut pas être de confiance pour protéger les données transmises entre les entités authentifiées par SASL. Dans une telle situation, une couche de sécurité SASL devrait être négociée qui ignore effectivement la présence de la sécurité de couche inférieure.

14.7 Technologies de mise en œuvre obligatoire

Au minimum, toutes les mises en œuvre DOIVENT prendre en charge les mécanismes suivants :
pour l'authentification : le mécanisme SASL [RFC2831]
pour la confidentialité : TLS (utilisant le chiffrement TLS_RSA_WITH_3DES_EDE_CBC_SHA)
pour les deux : TLS plus SASL EXTERNAL (en utilisant le chiffrement TLS_RSA_WITH_3DES_EDE_CBC_SHA qui prend en charge les certificats côté client)

14.8 Pare-feu

Les communications qui utilisent XMPP se font normalement sur des connexions de la [RFC0793] sur l'accès 5222 (client à serveur) ou l'accès 5269 (serveur à serveur) comme enregistré par l'IANA (voir à la Section 15 les considérations relatives à l'IANA)). L'utilisation de ces accès bien connus permet aux administrateurs d'activer ou désactiver facilement l'activité XMPP à travers les pare-feu existants et couramment déployés.

14.9 Utilisation de base64 dans SASL

Le client et le serveur DOIT tous deux vérifier toutes les données de la [RFC3548] reçue durant la négociation SASL. Une mise en œuvre DOIT rejeter (ne pas ignorer) tout caractère non explicitement permis par l'alphabet base64 ; cela aide à se garder contre la création d'un canal couvert qui pourrait être utilisé pour des "fuites" d'informations. Une mise en œuvre NE DOIT PAS caller sur une entrée invalide et DOIT rejeter toute séquence de caractères base64 qui contient le caractère bourrage ('=') si ce caractère est inclus comme quelque chose d'autre que le dernier caractère des données (par exemple, "=AAA" ou "BBBB=CCC") ; cela aide à se garder contre les attaques de débordement de mémoire tampon et autres attaques contre les mises en œuvre. Le codage base64 cache visuellement des informations autrement facilement reconnaissables, comme des mots de passe, mais ne fournit aucune confidentialité computationnelle. Le codage base64 DOIT suivre la définition de la Section 3 de la [RFC3548].

14.10 Profils Stringprep

XMPP utilise le profil [RFC3491] de la [RFC3454] pour le traitement des identifiants de domaine ; pour les considérations de sécurité relatives à Nameprep, se référer à la section appropriée de la [RFC3491].

En plus, XMPP définit deux profils de la [RFC3454] : Nodeprep (Appendice A) pour les identifiants de nœuds et Resourceprep (Appendice B) pour les identifiants de ressources.

Les répertoires Unicode et ISO/CEI 10646 ont de nombreux caractères qui paraissent similaires. Dans de nombreux cas, les utilisateurs de protocoles de sécurité peuvent faire une confrontation visuelle, comme lorsque on compare les noms de tiers de confiance. Parce qu'il est impossible de transposer les caractères qui paraissent similaires en dehors de tout contexte, comme de connaître la fonte utilisée, stringprep ne fait rien pour transposer ensemble les caractères qui paraissent similaires, ni pour interdire certains caractères parce qu'ils ressembleraient à d'autres.

Un identifiant de nœud peut être employé comme une partie de l'adresse d'une entité dans XMPP. Un usage courant est comme nom d'utilisateur d'un usager de messagerie instantanée ; une autre est comme nom d'une salle de débat multi utilisateurs ; de nombreuses autres sortes d'entités pourraient utiliser des identifiants de nœud comme partie de leur adresse. La sécurité de tels services pourrait être compromise sur la base des différentes interprétations de l'identifiant de nœud internationalisé ; par exemple, un usager qui entre un seul identifiant de nœud internationalisé pourrait accéder aux informations de compte d'un autre usager, ou un usager pourrait obtenir l'accès à une salle de débat ou service par ailleurs interdit.

Un identifiant de ressource peut être employé comme partie d'une adresse d'entité dans XMPP. Un usage courant est comme nom d'une ressource connectée d'utilisateur de messagerie instantanée (session active) ; un autre est comme surnom d'un usager dans une salle de débat multi utilisateur ; de nombreuses autres sortes d'entités pourraient utiliser des identifiants de ressource comme partie de leur adresse. La sécurité de tels services pourrait être compromise sur la base des différentes interprétations de l'identifiant de ressource internationalisé ; par exemple, un usager pourrait tenter d'initier plusieurs sessions avec le même nom, ou il pourrait envoyer un message à quelqu'un d'autre que le receveur désigné dans une salle de débat multi utilisateurs.

15. Considérations relatives à l'IANA

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

Un sous-espace de noms d'URN pour les données relatives à TLS 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-tls
Spécification : RFC 3920
Description : c'est le nom d'espace de noms XML pour les données relatives à TLS dans le protocole extensible de messagerie instantanée et de présence (XMPP) tel que défini par la RFC 3920.
Contact d'enregistrement : IETF, groupe de travail XMPP, < xmppwg@jabber.org >

15.2 Nom d'espace de noms XML pour les données SASL

Un sous-espace de noms d'URN pour les données relatives à SASL 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 la [RFC3688].)

URI : urn:ietf:params:xml:ns:xmpp-saslSpécification : RFC 3920
Description : c'est le nom d'espace de noms XML pour les données relatives à SASL dans le protocole extensible de messagerie instantanée et de présence (XMPP) tel que défini par la RFC 3920.
Contact d'enregistrement : IETF, groupe de travail XMPP, < xmppwg@jabber.org >

15.3 Nom d'espace de noms XML pour les erreurs de flux

Un sous-espace de noms d'URN pour les données d'erreur relatives aux flux 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 la [RFC3688].)

URI : urn:ietf:params:xml:ns:xmpp-streams
Spécification : RFC 3920
Description : c'est le nom d'espace de noms XML pour les données d'erreur relatives aux flux dans le protocole extensible de messagerie instantanée et de présence (XMPP) tel que défini par la RFC 3920.
Contact d'enregistrement : IETF, groupe de travail XMPP, < xmppwg@jabber.org >

15.4 Nom d'espace de noms XML pour les liens de ressource

Un sous-espace de noms d'URN pour les liens de ressource 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 la [RFC3688].)

URI : urn:ietf:params:xml:ns:xmpp-bind
Spécification : RFC 3920
Description : c'est le nom d'espace de noms XML pour les liens de ressource dans le protocole extensible de messagerie instantanée et de présence (XMPP) tel que défini par la RFC 3920.
Contact d'enregistrement : IETF, groupe de travail XMPP, < xmppwg@jabber.org >

15.5 Nom d'espace de noms XML pour les erreurs de strophe

Un sous-espace de noms d'URN pour les données d'erreur relatives aux strophes 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 la [RFC3688].)

URI : urn:ietf:params:xml:ns:xmpp-stanzas
Spécification : RFC 3920
Description : c'est le nom d'espace de noms XML pour les données d'erreur relatives aux strophes dans le protocole extensible de messagerie instantanée et de présence (XMPP) tel que défini par la RFC 3920.
Contact d'enregistrement : IETF, groupe de travail XMPP, < xmppwg@jabber.org >

15.6 Profil Nodeprep de Stringprep

Le profil Nodeprep de stringprep est défini sous Nodeprep (Appendice A). L'IANA a enregistré Nodeprep dans le registre de profil stringprep.

Nom de ce profil : Nodeprep
RFC dans laquelle le profil est défini : RFC 3920
Indication de si c'est ou non la version la plus récente du profil : c'est la première version de Nodeprep

15.7 Profil Resourceprep de Stringprep

Le profil Resourceprep de stringprep est défini sous Resourceprep (Appendice B). L'IANA a enregistré Resourceprep dans le registre du profil stringprep.

Nom de ce profil : Resourceprep
RFC dans laquelle le profil est défini : RFC 3920
Indication de si c'est ou non la version la plus récente du profil : c'est la première version de Nodeprep

15.8 Nom de service GSSAPI

L'IANA a enregistré "xmpp" comme nom de service GSSAPI [RFC2743], comme défini a paragraphe 6.3 "Définition SASL".

15.9 Numéros d'accès

L'IANA a enregistré "xmpp-client" et "xmpp-server" comme mots clés pour respectivement les accès 5222 et 5269 [RFC0793].

Ces accès DEVRAIENT être utilisés pour les communications respectivement de client à serveur et de serveur à serveur, mais leur utilisation est FACULTATIVE.

16. Références

16.1 Références normatives

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

[RFC1035] P. Mockapetris, "Noms de domaines – Mise en œuvre et spécification", STD 13, novembre 1987. (*MàJ par la* RFC6604)

[RFC1750] D. Eastlake 3rd et autres, "Recommandations d'aléa pour la sécurité", décembre 1994. *(Info., remplacée par la* RFC4086)

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

[RFC2222] J. Myers, "Authentification simple et couche de sécurité (SASL)", octobre 1997. (*Obsolète, voir* RFC4422, RFC4752) (*MàJ par* RFC2444) *(P.S.)*

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", novembre 1997. (*Obsolète, voir* RFC5234)

[RFC2246] T. Dierks et C. Allen, "Protocole TLS version 1.0", janvier 1999.

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

[RFC2743] J. Linn, "Interface générique de programme d'application de service de sécurité, version 2, mise à jour 1", janvier 2000. (*MàJ par* RFC5554)

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

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

[RFC2831] P. Leach et C. Newman, "Utilisation de l'authentification par résumé comme mécanisme SASL", mai 2000. (*Obsolète, voir* RFC6331)

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

[RFC3280] R. Housley, W. Polk, W. Ford et D. Solo, "Profil de certificat d'infrastructure de clé publique X.509 et de liste de révocation de certificat (CRL) pour l'Internet", avril 2002. *(Obsolète, voir* RFC5280)

[RFC3454] P. Hoffman et M. Blanchet, "Préparation de chaînes internationalisées ("stringprep")", décembre 2002. *(P.S.)*

[RFC3490] P. Faltstrom et autres, "Internationalisation des noms de domaine dans les applications (IDNA)", mars 2003.

(*Remplacée par les* RFC5890 et 5891, *P.S.*)

[RFC3491] P. Hoffman et M. Blanchet, "Nameprep : Profil Stringprep pour les noms de domaine internationalisés (IDN)", mars 2003. (*Remplacée par la* RFC5891, *P.S.*)

[RFC3513] R. Hinden et S. Deering, "Architecture d'adressage du protocole Internet version 6 (IPv6)", avril 2003. (*Obs. voir* RFC4291)

[RFC3548] S. Josefsson, "Codages de données Base16, Base32, et Base64", juillet 2003. *(Obsolète, voir* 4648) (Info)

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

[UCS2] Organisation Internationale de Normalisation, "Technologie de l'information – Jeu de caractères universel codé sur plusieurs octets (UCS) - Amendement 2 : Format n° 8 de transformation d'UCS (UTF-8)", Norme ISO 10646-1 Addendum 2, octobre 1996.

[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

[ASN.1] Recommandation UIT-T X.208: "Spécification de la Notation numéro 1 de syntaxe abstraite (ASN.1)", 1988.

[JEP-0029] Kaes, C., "Definition of Jabber Identifiers (JID)", JSF JEP 0029, octobre 2003.

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

[JEP-0086] Norris, R. et P. Saint-Andre, "Error Condition Mappings", JSF JEP 0086, février 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)*

[RFC1939] J. Myers, M. Rose, "Protocole Post Office - version 3", mai 1996. (*MàJ par* RFC1957, RFC2449) (STD0053)

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

[RFC2393] A. Shacham et autres, "Protocole de compression de charge utile IP (IPComp)", décembre 1998. (*Obsolète, voir* RFC3173)

[RFC2535] D. Eastlake, 3rd, "Extensions de sécurité du système des noms de domaines", mars 1999. (*Obsolète, voir* RFC4033, RFC4034, RFC4035) *(P.S.)*

[RFC2595] C. Newman, "Utilisation de TLS avec IMAP, POP3 et ACAP", juin 1999. (*MàJ par* RFC4616) *(P.S.)*

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

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

[RFC2821] J. Klensin, éditeur, "Protocole simple de transfert de messagerie", STD 10, avril 2001. *(Obsolète, voir* RFC5321)

[RFC3501] M. Crispin, "Protocole d'accès au message Internet - version 4rev1", mars 2003. (*MàJ par* RFC4466, RFC4469, RFC4551, RFC5032, RFC5182) *(P.S.)*

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

[RFC3921] P. Saint-Andre, éd., "Protocole extensible de messagerie et de présence (XMPP) : messagerie instantanée et présence", octobre 2004. *(P.S.)*

[SIMPLE] Groupe de travail SIMPLE, "SIMPLE WG", < http://www.ietf.org/html.charters/simple-charter.html >.

Appendice A. Nodeprep

A.1 Introduction

Le présent appendice définit le profil "Nodeprep" de la [RFC3454]. À ce titre, il spécifie les règles de traitement qui vont permettre aux utilisateurs d'entrer des identifiants de nœud internationalisés dans le protocole extensible de messagerie instantanée et de présence (XMPP) et d'avoir les meilleures chances d'obtenir le contenu correct des chaînes. (Un identifiant de nœud XMPP est la portion facultative d'une adresse XMPP qui précède un identifiant de domaine et le séparateur '@' ; il est souvent, mais pas exclusivement, associé à un nom d'utilisateur de messagerie instantanée.) Ces règles de traitement ne sont destinées qu'aux identifiants de nœud XMPP et ne sont pas destinées à du texte arbitraire ou à tout autre aspect d'une adresse XMPP.

Ce profil définit ce qui suit, comme exigé par la [RFC3454] :
- L'applicabilité du profil : identifiants de ressource internationalisés au sein de XMPP
- Le répertoire de caractères qui est l'entrée à, et le résultat de stringprep : Unicode 3.2, spécifié à la Section 2 de cet appendice
- Les transpositions utilisées : spécifié à la Section 3
- La normalisation Unicode utilisée : spécifié à la Section 4
- Les caractères interdits en sortie : spécifié à la Section 5
- Le traitement des caractères bidirectionnels : spécifié à la Section 6

A.2 Répertoire de caractères

Ce profil utilise Unicode 3.2 avec la liste des codets non alloués du Tableau A.1, définis à l'Appendice A de la [RFC3454].

A.3 Transposition

Ce profil spécifie la transposition en utilisant les tableaux suivants de la [RFC3454] :
Tableau B.1
Tableau B.2

A.4 Normalisation

Ce profil spécifie l'utilisation de la forme Unicode de normalisation KC, comme décrit à la [RFC3454].

A.5 Résultats interdits

Ce profil spécifie l'interdiction d'utilisation des tableaux suivants de la [RFC3454].
Tableau C.1.1
Tableau C.1.2
Tableau C.2.1
Tableau C.2.2
Tableau C.3
Tableau C.4
Tableau C.5
Tableau C.6
Tableau C.7
Tableau C.8
Tableau C.9

De plus, les caractères Unicode suivants sont aussi interdits : #x22 ("), #x26 (&), #x27 (') ,#x2F (/), #x3A (:), #x3C (<), #x3E (>), x40 (@)

A.6 Caractères bidirectionnels

Ce profil spécifie la vérification des chaînes bidirectionnelles, comme décrit à la Section 6 de la [RFC3454].

Appendice B. Resourceprep

B.1 Introduction

Cet appendice définit le profil "Resourceprep" de la [RFC3454]. À ce titre, il spécifie les règles de traitement qui vont permettre aux utilisateurs d'entrer des identifiants de ressource internationalisés dans le protocole extensible de messagerie instantanée et de présence (XMPP, *Extensible Messaging et Presence Protocol*) et d'avoir les plus fortes chances que le contenu des chaînes soit correct. (Un identifiant de ressource XMPP est la portion facultative d'une adresse XMPP qui suit un identifiant de domaine et le séparateur '/' ; il est souvent mais pas exclusivement associé à un nom de session de messagerie instantanée.) Ces règles de traitement ne sont destinées qu'aux identifiants de ressource XMPP et ne sont pas destinés à du texte arbitraire ou à tout autre aspect d'une adresse XMPP.

Ce profil définit ce qui suit, comme requis par la [RFC3454]:
- L'applicabilité du profil : identifiants de ressource internationalisés au sein de XMPP
- Le répertoire de caractères qui est l'entrée à, et le résultat de stringprep : Unicode 3.2, spécifié à la Section 2 de cet appendice
- Les transpositions utilisées : spécifié à la Section 3
- La normalisation Unicode utilisée : spécifié à la Section 4
- Les caractères interdits en sortie : spécifié à la Section 5
- Le traitement des caractères bidirectionnels : spécifié à la Section 6

B.2 Répertoire de caractères

Ce profil utilise Unicode 3.2 avec la liste des codets non alloués du Tableau A.1, défini à l'Appendice A de la [RFC3454].

B.3 Transposition

Ce profil spécifie une transposition en utilisant le tableau suivant de la [RFC3454] : Tableau B.1

B.4 Normalisation

Ce profil spécifie l'utilisation de la forme KC de normalisation Unicode, comme décrit dans la [RFC3454].

B.5 Résultat interdit

Ce profil spécifie l'interdiction de l'utilisation des tableaux suivants de la [RFC3454].
Tableau C.1.2
Tableau C.2.1
Tableau C.2.2
Tableau C.3
Tableau C.4
Tableau C.5
Tableau C.6
Tableau C.7
Tableau C.8
Tableau C.9

B.6 Caractères bidirectionnels

Ce profil spécifie la vérification des chaînes bidirectionnelles comme décrit à la Section 6 de la [RFC3454].

Appendice C Schémas XML

Les schémas XML suivants sont descriptifs et non normatifs. Pour les schémas qui définissent les espaces de noms 'jabber:client' et 'jabber:server', se référer à la [RFC3921].

C.1 Espace de noms de flux

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

<xs:schema
   xmlns:xs='http://www.w3.org/2001/XMLSchema'
   targetNamespace='http://etherx.jabber.org/streams'
   xmlns='http://etherx.jabber.org/streams'
   elementFormDefault='unqualified'>

 <xs:element name='stream'>
  <xs:complexType>
   <xs:sequence xmlns:client='jabber:client'
                xmlns:server='jabber:server'
                xmlns:db='jabber:server:dialback'>
    <xs:element ref='features' minOccurs='0' maxOccurs='1'/>
    <xs:any namespace='urn:ietf:params:xml:ns:xmpp-tls'
          minOccurs='0'
          maxOccurs='unbounded'/>
    <xs:any namespace='urn:ietf:params:xml:ns:xmpp-sasl'
          minOccurs='0'
          maxOccurs='unbounded'/>
    <xs:choice minOccurs='0' maxOccurs='1'>
     <xs:choice minOccurs='0' maxOccurs='unbounded'>
      <xs:element ref='client:message'/>
      <xs:element ref='client:presence'/>
      <xs:element ref='client:iq'/>
     </xs:choice>
     <xs:choice minOccurs='0' maxOccurs='unbounded'>
      <xs:element ref='server:message'/>
      <xs:element ref='server:presence'/>
      <xs:element ref='server:iq'/>
      <xs:element ref='db:result'/>
      <xs:element ref='db:verify'/>
     </xs:choice>
    </xs:choice>
    <xs:element ref='error' minOccurs='0' maxOccurs='1'/>
   </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='version' type='xs:decimal' use='optional'/>
   <xs:attribute ref='xml:lang' use='optional'/>
  </xs:complexType>
 </xs:element>

 <xs:element name='features'>
  <xs:complexType>
   <xs:all xmlns:tls='urn:ietf:params:xml:ns:xmpp-tls'
         xmlns:sasl='urn:ietf:params:xml:ns:xmpp-sasl'
         xmlns:bind='urn:ietf:params:xml:ns:xmpp-bind'
         xmlns:sess='urn:ietf:params:xml:ns:xmpp-session'>
    <xs:element ref='tls:starttls' minOccurs='0'/>
    <xs:element ref='sasl:mechanisms' minOccurs='0'/>
    <xs:element ref='bind:bind' minOccurs='0'/>
    <xs:element ref='sess:session' minOccurs='0'/>
   </xs:all>
  </xs:complexType>
 </xs:element>

 <xs:element name='error'>
  <xs:complexType>
   <xs:sequence  xmlns:err='urn:ietf:params:xml:ns:xmpp-streams'>
```

```
      <xs:group   ref='err:streamErrorGroup'/>
      <xs:element ref='err:text'
                  minOccurs='0'
                  maxOccurs='1'/>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

C.2 Espace de noms d'erreur de flux

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-streams'
    xmlns='urn:ietf:params:xml:ns:xmpp-streams'
    elementFormDefault='qualified'>
  <xs:element name='bad-format' type='empty'/>
  <xs:element name='bad-namespace-prefix' type='empty'/>
  <xs:element name='conflict' type='empty'/>
  <xs:element name='connection-timeout' type='empty'/>
  <xs:element name='host-gone' type='empty'/>
  <xs:element name='host-unknown' type='empty'/>
  <xs:element name='improper-addressing' type='empty'/>
  <xs:element name='internal-server-error' type='empty'/>
  <xs:element name='invalid-from' type='empty'/>
  <xs:element name='invalid-id' type='empty'/>
  <xs:element name='invalid-namespace' type='empty'/>
  <xs:element name='invalid-xml' type='empty'/>
  <xs:element name='not-authorized' type='empty'/>
  <xs:element name='policy-violation' type='empty'/>
  <xs:element name='remote-connection-failed' type='empty'/>
  <xs:element name='resource-constraint' type='empty'/>
  <xs:element name='restricted-xml' type='empty'/>
  <xs:element name='see-other-host' type='xs:string'/>
  <xs:element name='system-shutdown' type='empty'/>
  <xs:element name='undefined-condition' type='empty'/>
  <xs:element name='unsupported-encoding' type='empty'/>
  <xs:element name='unsupported-stanza-type' type='empty'/>
  <xs:element name='unsupported-version' type='empty'/>
  <xs:element name='xml-not-well-formed' type='empty'/>

  <xs:group name='streamErrorGroup'>
    <xs:choice>
      <xs:element ref='bad-format'/>
      <xs:element ref='bad-namespace-prefix'/>
      <xs:element ref='conflict'/>
      <xs:element ref='connection-timeout'/>
      <xs:element ref='host-gone'/>
      <xs:element ref='host-unknown'/>
      <xs:element ref='improper-addressing'/>
      <xs:element ref='internal-server-error'/>
      <xs:element ref='invalid-from'/>
      <xs:element ref='invalid-id'/>
      <xs:element ref='invalid-espace de noms'/>
      <xs:element ref='invalid-xml'/>
      <xs:element ref='not-authorized'/>
      <xs:element ref='policy-violation'/>
      <xs:element ref='remote-connection-failed'/>
      <xs:element ref='resource-constraint'/>
      <xs:element ref='restricted-xml'/>
```

```
      <xs:element ref='see-other-host'/>
      <xs:element ref='system-shutdown'/>
      <xs:element ref='undefined-condition'/>
      <xs:element ref='unsupported-encoding'/>
      <xs:element ref='unsupported-stanza-type'/>
      <xs:element ref='unsupported-version'/>
      <xs:element ref='xml-not-well-formed'/>
    </xs:choice>
  </xs:group>

  <xs:element name='text'>
    <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:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

C.3 Espace de noms TLS

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

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-tls'
    xmlns='urn:ietf:params:xml:ns:xmpp-tls'
    elementFormDefault='qualified'>
  <xs:element name='starttls'>
    <xs:complexType>
      <xs:sequence>
        <xs:element
            name='required'
            minOccurs='0'
            maxOccurs='1'
            type='empty'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='proceed' type='empty'/>
  <xs:element name='failure' type='empty'/>
  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

C.4 Espace de noms SASL

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

<xs:schema
```

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

  <xs:element name='mechanisms'>
    <xs:complexType>
      <xs:sequence>
       <xs:element name='mechanism'
                   maxOccurs='unbounded'
                   type='xs:string'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='auth'>
   <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='empty'>
         <xs:attribute name='mechanism'
                       type='xs:string'
                       use='optional'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name='challenge' type='xs:string'/>
  <xs:element name='response' type='xs:string'/>
  <xs:element name='abort' type='empty'/>
  <xs:element name='success' type='empty'/>

  <xs:element name='failure'>
    <xs:complexType>
      <xs:choice minOccurs='0'>
        <xs:element name='aborted' type='empty'/>
        <xs:element name='incorrect-encoding' type='empty'/>
        <xs:element name='invalid-authzid' type='empty'/>
        <xs:element name='invalid-mechanism' type='empty'/>
        <xs:element name='mechanism-too-weak' type='empty'/>
        <xs:element name='not-authorized' type='empty'/>
        <xs:element name='temporary-auth-failure' type='empty'/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name='empty'>
    <xs:restriction base='xs:string'>
      <xs:enumeration value=''/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

C.5 Espace de noms de lien de ressource

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

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

```
  <xs:element name='bind'>
   <xs:complexType>
    <xs:choice minOccurs='0' maxOccurs='1'>
     <xs:element name='resource' type='xs:string'/>
      <xs:element name='jid' type='xs:string'/>
    </xs:choice>
   </xs:complexType>
  </xs:element>

</xs:schema>
```

C.6 Espace de noms de rappel

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

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

  <xs:element name='result'>
   <xs:complexType>
    <xs:simpleContent>
     <xs:extension base='xs:token'>
      <xs:attribute name='from' type='xs:string' use='required'/>
      <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='invalid'/>
         <xs:enumeration value='valid'/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
     </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
  </xs:element>

  <xs:element name='verify'>
   <xs:complexType>
    <xs:simpleContent>
     <xs:extension base='xs:token'>
      <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='optional'>
       <xs:simpleType>
        <xs:restriction base='xs:NCName'>
         <xs:enumeration value='invalid'/>
         <xs:enumeration value='valid'/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
     </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
  </xs:element>

</xs:schema>
```

C.7 Espace de noms d'erreur de strophe

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

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

 <xs:element name='bad-request' type='empty'/>
 <xs:element name='conflict' type='empty'/>
 <xs:element name='feature-not-implemented' type='empty'/>
 <xs:element name='forbidden' type='empty'/>
 <xs:element name='gone' type='xs:string'/>
 <xs:element name='internal-server-error' type='empty'/>
 <xs:element name='item-not-found' type='empty'/>
 <xs:element name='jid-malformed' type='empty'/>
 <xs:element name='not-acceptable' type='empty'/>
 <xs:element name='not-allowed' type='empty'/>
 <xs:element name='payment-required' type='empty'/>
 <xs:element name='recipient-unavailable' type='empty'/>
 <xs:element name='redirect' type='xs:string'/>
 <xs:element name='registration-required' type='empty'/>
 <xs:element name='remote-server-not-found' type='empty'/>
 <xs:element name='remote-server-timeout' type='empty'/>
 <xs:element name='resource-constraint' type='empty'/>
 <xs:element name='service-unavailable' type='empty'/>
 <xs:element name='subscription-required' type='empty'/>
 <xs:element name='undefined-condition' type='empty'/>
 <xs:element name='unexpected-request' type='empty'/>

 <xs:group name='stanzaErrorGroup'>
  <xs:choice>
   <xs:element ref='bad-request'/>
   <xs:element ref='conflict'/>
   <xs:element ref='feature-not-implemented'/>
   <xs:element ref='forbidden'/>
   <xs:element ref='gone'/>
   <xs:element ref='internal-server-error'/>
   <xs:element ref='item-not-found'/>
   <xs:element ref='jid-malformed'/>
   <xs:element ref='not-acceptable'/>
   <xs:element ref='not-allowed'/>
   <xs:element ref='payment-required'/>
   <xs:element ref='recipient-unavailable'/>
   <xs:element ref='redirect'/>
   <xs:element ref='registration-required'/>
   <xs:element ref='remote-server-not-found'/>
   <xs:element ref='remote-server-timeout'/>
    <xs:element ref='resource-constraint'/>
   <xs:element ref='service-unavailable'/>
   <xs:element ref='subscription-required'/>
   <xs:element ref='undefined-condition'/>
   <xs:element ref='unexpected-request'/>
  </xs:choice>
 </xs:group>

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

</xs:schema>
```

Appendice D. Différences entre les cœurs de protocoles nJabber et XMPP

Cette section n'est pas normative.

XMPP a été adapté des protocoles originellement développés dans la communauté libre Jabber, qui peuvent être vus comme "XMPP 0.9". Parce qu'il existe une large base installée de mises en œuvre et déploiements Jabber, il peut être utile de spécifier les différences clés entre les protocoles Jabber pertinents et XMPP afin d'accélérer et encourager les mises à niveau de ces mises en œuvre et déploiements en XMPP. Cette section résume le cœur des différences, tandis que les paragraphes correspondants de la [RFC3921] résument les différences qui se rapportent spécifiquement aux applications de messagerie instantanée et de présence.

D.1 Chiffrement de canal

Il était de pratique courante dans la communauté Jabber d'utiliser SSL pour le chiffrement de canal sur des accès autres que 5222 et 5269 (la convention est d'utiliser les accès 5223 et 5270). XMPP utilise TLS sur les accès enregistrés par l'IANA pour le chiffrement de canal, comme défini ici à la Section 5 "Utilisation de TLS".

D.2 Authentification

Le protocole d'authentification client-serveur développé dans la communauté Jabber utilisait une interaction IQ de base qualifiée par l'espace de noms 'jabber:iq:auth' (la documentation de ce protocole est contenue dans [JEP-0078], publié par la Fondation de logiciel Jabber (JBS, *Jabber Software Foundation*) [JSF]. XMPP utilise SASL pour l'authentification, comme défini ici à la Section 6 "Utilisation de SASL".

La communauté Jabber n' pas développé de protocole d'authentification pour les communications de serveur à serveur, mais seulement le protocole de rappel de serveur (Section 8) pour empêcher l'usurpation de serveur. XMPP se substitue au rappel de serveur avec un vrai protocole d'authentification de serveur à serveur, comme défini ici à la Section 6 "Utilisation de SASL".

D.3 Lien de ressource

Le lien de ressource était traité dans la communauté Jabber via l'espace de noms 'jabber:iq:auth' (qui était aussi utilisé pour l'authentification d'un client auprès d'un serveur). XMPP définit un espace de noms dédié pour le lien de ressource ainsi que la capacité pour un serveur de générer un identifiant de ressource au nom d'un client, comme défini à la Section 7 "Lien de ressource".

D.4 Traitement de JID

Le traitement de JID était défini de façon assez lâche par la communauté Jabber (la documentation des caractères interdits et du traitement de la casse est contenu dans [JEP-0029], publié par la Fondation de logiciels Jabber [JSF]). XMPP spécifie l'utilisation de la [RFC3491] pour les identifiants de domaines et complète Nameprep avec deux profils additionnels [RFC3454] pour le traitement des JID : Nodeprep (Appendice A) pour les identifiants de nœuds et Resourceprep (Appendice B) pour les identifiants de ressources.

D.5 Traitement des erreurs

Les erreurs relatives aux flux étaient traitées dans la communauté Jabber via un texte XML de données de caractères dans un élément <stream:error/>. Dans XMPP, les erreurs relatives aux flux sont traitées via un mécanisme extensible défini au paragraphe 4.7 "Erreurs de flux".

Les erreurs relatives aux strophes étaient traitées dans la communauté Jabber via des codes d'erreur de style HTTP. Dans XMPP, les erreurs relatives aux strophes sont traitées via un mécanisme extensible défini au paragraphe 9.3 "Erreurs de strophes". (La documentation de la transposition entre les mécanismes de traitement d'erreur Jabber et XMPP est contenue dans [JEP-0086], publié par la Fondation du logiciel Jabber [JSF].)

D.6 Internationalisation

Bien que l'utilisation de UTF-8 ait toujours été de pratique standard au sein de la communauté Jabber, la communauté n'a pas défini de mécanisme pour spécifier le langage des textes lisibles par l'homme fournis dans les données de caractères XML. XMPP spécifie l'utilisation de l'attribut 'xml:lang' dans de tels contextes, comme défini au paragraphe 4.4 "Attributs de flux" et xml:lang (paragraphe 9.1.5).

D.7 Attribut de version de flux

La communauté Jabber n'a pas inclus d'attribut 'version' dans les en-têtes de flux. XMPP spécifie l'inclusion de cet attribut comme moyen de signaler la prise en charge des caractéristiques du flux (authentification, chiffrement, etc.) défini ici au paragraphe 4.4.1 "Prise en charge de version".

Contributeurs

La plupart des aspects du cœur 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 de la version initiale du serveur jabber en janvier 1999. Les premiers contributeurs majeurs du protocole de base incluaient aussi Ryan Eatmon, Peter Millard, Thomas Muldowney, et Dave Smith. Le travail du groupe XMPP s'est concentré en particulier sur la sécurité et l'internationalisation ; dans ces domaines, les protocoles pour l'utilisation de TLS et SASL ont été apportés à l'origine par Rob Norris, et les profils stringprep ont été apportés par Joe Hildebrand. La syntaxe de code d'erreur a été suggérée par Lisa Dusseault.

Remerciements

Des remerciement sont dus à de nombreuses personnes en plus de la liste des contributeurs. Bien qu'il soit difficile d'en donner une liste complète, les personnes suivantes ont été d'une aide particulière pour la définition des protocoles ou en commentant les spécifications du présent mémoire : Thomas Charron, Richard Dobson, Sam Hartman, Schuyler Heath, Jonathan Hogg, Cullen Jennings, Craig Kaes, Jacek Konieczny, Alexey Melnikov, Keith Minkler, Julian Missig, Pete Resnick, Marshall Rose, Alexey Shchepin, Jean-Louis Seguineau, Iain Shigeoka, Greg Troxel, et David Waite. Merci aussi aux membres du groupe de travail XMPP et à la communauté de l'IETF pour leurs commentaires et retours produits pendant toute la vie de ce mémoire.

Adresse de l'auteur

Peter Saint-Andre (éditeur)
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