Groupe de travail Réseau
**Request for Comments : 2824**
Catégorie : Information
Traduction Claude Brière de L'Isle

J. Lennox, Columbia University
H. Schulzrinne, Columbia University
mai 2000

# Cadre et exigences du langage de traitement d'appel



**Résumé**
Un grand nombre des services qu'on souhaite rendre possibles pour la téléphonie Internet exigent que s'achèvent des combinaisons souvent très élaborées d'opérations de signalisation dans les appareils du réseau. On veut une façon simple et normalisée pour créer de tels services et les rendre plus faciles à mettre en œuvre et déployer. Le présent document décrit le cadre architectural d'un tel mécanisme, que nous appelons un langage de traitement d'appel. Il décrit aussi les exigences pour un tel langage.

## Table des Matières

## 1. Introduction

Récemment, plusieurs protocoles ont été créés pour permettre d'effectuer des appels téléphoniques sur les réseaux IP, notamment SIP [RFC2543] et [H.323]. Ces normes émergentes ont ouvert la possibilité d'une large et importante décentralisation de la fourniture des services de téléphonie de sorte qu'ils puissent être sous le contrôle de l'usager.

De nombreux services de téléphonie Internet peuvent être mis en œuvre, et le seront, entièrement sur des appareils d'extrémité. Des appels multi parties, par exemple, ou des tonalités d'alerte d'appel en attente, ou des services de mise en attente, dépendent largement de l'état du système d'extrémité et du contenu spécifique des flux du support, informations qui souvent ne sont disponibles qu'au système d'extrémité. Divers services, cependant – ceux qui impliquent la localisation de l'usager, la distribution des appels, le comportement en cas d'occupation des systèmes d'extrémité, et ainsi de suite – sont indépendants de l'appareil d'extrémité particulier, ou ont besoin d'être opérationnels même lorsque un appareil d'extrémité est indisponible. Ces services sont quand même mieux situés dans un appareil réseau, plutôt que dans le système d'extrémité.

Traditionnellement, les services fondés sur le réseau n'ont été créés que par les fournisseurs de services. La création de service impliquait normalement d'utiliser des outils brevetés ou soumis à restrictions, et il y avait peu de place pour la personnalisation ou l'amélioration par les utilisateurs finaux. Cependant, dans l'environnement de l'Internet, ceci est en train de changer. La connexité mondiale et les protocoles ouverts permettent aux utilisateurs finaux ou aux tiers de concevoir et mettre en œuvre des services nouveaux ou personnalisés, et de déployer et modifier de façon dynamique leur service sans avoir besoin qu'un fournisseur de service agisse comme intermédiaire.

Un certain nombre d'applications Internet ont un tel environnement personnalisé – par exemple, la Toile a CGI [RFC3875], et la messagerie électronique a Sieve [RFC3028] ou procmail. Pour créer un tel environnement ouvert et personnalisé pour la téléphonie Internet, on a besoin d'un moyen sûr et normalisé pour que ces créateurs de nouveaux services décrivent le comportement attendu des serveurs du réseau.

Le présent document décrit une architecture dans laquelle les appareils du réseau répondent aux événements de signalisation d'appel en déclenchant des programmes créés par l'usager écrits dans un langage simple, statique, non expressivement complet. On appelle ce langage un langage de traitement d'appel.

Le développement du présent document a été substantiellement nourri par le développement d'un langage de traitement d'appel particulier, décrit dans la [RFC3880]. En général, quand le présent document se réfère à un "langage de traitement d'appel", il vise un langage générique qui remplit ce rôle ; "le langage de traitement d'appel" ou "le CPL" se réfère à ce langage particulier.

## 2. Terminologie

Dans cette section, on définit certains des termes utilisés dans le présent document.

La terminologie SIP [RFC2543] utilisée inclut :

invitation : demande INVITE initiale d'une transaction SIP, par laquelle une partie initie un appel avec une autre.

serveur de redirection : appareil SIP qui répond aux invitations et autres demandes en informant l'origine de la demande d'une adresse de remplacement à laquelle la demande devrait être envoyée.

serveur mandataire : appareil SIP qui reçoit les invitations et autres demandes, et les transmet aux autres appareils SIP. Il reçoit ensuite les réponses aux demandes qu'il a transmises, et les retransmet à l'envoyeur de la demande initiale.

agent d'utilisateur : appareil SIP qui crée et reçoit les demandes, afin d'établir ou affecter autrement l'état d'un appel. Cela peut être, par exemple, un téléphone ou un système de messagerie vocale.

client d'agent d'utilisateur : portion d'un agent d'utilisateur qui initie les demandes.

serveur d'agent d'utilisateur : portion d'un agent d'utilisateur qui répond aux demandes.

La terminologie de [H.323] utilisée inclut :

terminal : appareil H.323 qui génère et reçoit des appels, et leur support associé.

portier : entité H.323 sur le réseau qui assure la traduction d'adresse et contrôle l'accès au réseau pour les terminaux H.323 et autres points d'extrémité. Le portier peut aussi fournir d'autres services aux points d'extrémité tels que la gestion de la bande passante et la localisation des routeurs.

routeur : appareil qui traduit les appels entre un réseau H.323 et un autre réseau, normalement le réseau téléphonique public commuté.

RAS : messages d'enregistrement, admission et état communiqués entre deux entités H.323, par exemple entre un point d'extrémité et un portier.

Terminologie générale utilisée dans le présent document :

localisation d'utilisateur : processus par lequel un appareil de téléphone Internet détermine où on peut trouver un usager désigné par une certaine adresse.

CPL *(Call Processing Language)* : langage simple pour décrire comment devraient être traitées les invitations d'appel de téléphonie Internet.

script : instance particulière d'un CPL, décrivant un ensemble particulier de services désirés.

système d'extrémité : appareil à partir duquel et vers lequel des appels sont établis. Il crée et reçoit les supports d'appels (audio, vidéo, ou autres). Cela peut être un agent d'utilisateur SIP ou un terminal H.323.

serveur de signalisation : appareil qui traite l'acheminement des invitations d'appel. Il ne traite pas ni n'interagit avec les supports d'un appel. Il peut être un mandataire SIP ou un serveur de redirection, ou un portier H.323.

## 3. Exemple de services

Pour motiver la discussion qui suit, la présente section donne des exemples spécifiques de services dont on veut que les usagers soient capables de les créer par programme. Noter que certains de ces exemples sont délibérément un peu compliqués, afin de démontrer le niveau de logique de décision qui devrait être possible.

o Transfert d'appel sur occupation/non réponse

Lorsque un nouvel appel arrive, cela devrait sonner sur le téléphone de l'usager. Si il est occupé, l'appel devrait toujours être redirigé sur la boîte aux lettres vocale de l'usager. Si, au lieu de cela, il n'y a pas de réponse après quatre sonneries, il devrait aussi être redirigé sur sa messagerie vocale, sauf si il provient d'un superviseur, auquel cas il devrait être mandaté au téléphone cellulaire de l'usager si celui-ci est bien enregistré.

o Adresse d'informations

Une société annonce une adresse générale "d'informations" pour les consommateurs potentiels. Lorsque un appel arrive à cette adresse, si c'est pendant les heures ouvrables, l'appelant devrait se voir communiquer une liste de personnes qui acceptent à ce moment les appels d'informations générales. Si c'est en dehors des heures de travail, l'appelant devrait obtenir une page d'accueil qui lui indique les heures où il peut appeler.

o Localisation intelligente d'usager

Lorsque un appel arrive, on devrait consulter la liste des localisations où l'usager s'est enregistré. Selon le type d'appel (travail, personnel, etc.) l'appel devrait sonner sur un sous-ensemble approprié des localisations enregistrées, selon les informations contenues dans les enregistrements. Si l'usager décroche de plus d'une station, les décrochages devraient être rapportés séparément à l'appelant.

o Localisation intelligente de l'usager avec connaissance du support

Lorsqu'un appel arrive, il devrait être mandaté à la station que l'usager a enregistrée avec les capacités de support qui correspondent le mieux à celles spécifiées dans la demande d'appel. Si l'usager ne décroche pas de cette station dans l'intervalle de quatre sonneries, l'appel devrait être renvoyé aux autres stations sur lesquelles il s'est enregistré, en suivant l'ordre décroissant de correspondance des capacités.

,

o Affectation à la facture du client – bureau d'avocat

Lorsque l'appel arrive, l'adresse de l'appelant est corrélée avec le client correspondant, son nom et son adresse, et l'heure à laquelle l'appel est enregistré. Si on ne trouve pas de client qui corresponde, l'appel est transmis au secrétariat.

## 4. Scénarios d'utilisation

Un CPL serait utile pour mettre en œuvre des services dans un certain nombre de scénarios différents.

o Création de script par l'utilisateur final
Dans l'approche la plus directe de la création d'un service avec un CPL, un utilisateur crée simplement un script qui décrit son service. Il décide simplement quel service il veut, le décrit en utilisant un script CPL, et en le chargeant ensuite sur un serveur.

o Fourniture par un tiers
Comme un CPL est un langage normalisé, il peut aussi être utilisé pour permettre à des tiers de créer des services personnalisés pour les clients. Ces scripts peuvent alors fonctionner sur des serveurs que possèdent les utilisateurs finaux ou par le fournisseur de service de l'utilisateur.

o Définition du service d'administrateur
Un CPL peut aussi être utilisé par des administrateurs de serveur pour créer des services simples ou décrire une politique pour les serveurs sous leur contrôle. Si un serveur met en œuvre des services en CPL, dans tous les cas, l'extension de l'architecture de service pour permettre aux administrateurs comme aux usagers de créer des scripts est une simple extension.

o Logiciels sur la Toile
Finalement, il y a eu un certain nombre de propositions de création ou de personnalisation de services à l'aide des interfaces de la Toile. Un CPL pourrait être utilisé comme arrière plan pour de tels environnements : une application de la Toile pourrait créer un script en CPL au nom d'un usager, et le serveur de téléphonie pourrait alors mettre en œuvre les services sans qu'aucun composant doive être au courant des spécificités de l'autre.

## 5. Création de CPL

Les scripts en CPL pourraient aussi être créés par divers moyens. Comme HTML, qui peut être créé de nombreuses façons différentes, on envisage de multiples styles de création de scripts CPL.

o Création manuelle
Le plus directement, des scripts CPL peuvent être créés à la main, par des utilisateurs qui s'y connaissent. Le CPL décrit dans la [RFC3880] a un format de texte avec une syntaxe sans complications, de sorte que la création manuelle sera directe.

o Scripts automatisés
Des dispositifs en CPL peuvent être créés par des moyens automatiques, comme dans l'exemple des logiciels sur la Toile décrits au paragraphe précédent. Avec une syntaxe simple, fondée sur le texte, les langages de traitement de texte standard seront capables de créer et éditer facilement des scripts CPL.

o Outils GUI
Finalement, les utilisateurs seront capables d'utiliser les outils d'interface graphique d'usager (GUI, *Graphical User Interface)* pour créer et éditer les scripts CPL. On s'attend à ce que la plupart des utilisateurs moyens vont suivre cette approche une fois que le CPL se sera répandu. Le CPL sera conçu avec cette application comme objectif, afin que la pleine puissance d'expression des scripts puisse être représentée simplement et directement d'une manière graphique.

## 6. Modèle de réseau

Le langage de traitement d'appel fonctionne sur un modèle généralisé de réseau téléphonique Internet. Bien que les détails des divers protocoles diffèrent, à un niveau abstrait, toutes les architectures majeures de téléphonie Internet sont suffisamment similaires pour que leurs caractéristiques majeures puissent être décrites en commun. Le présent document utilise généralement la terminologie SIP, car c'est avec ce protocole que les auteurs ont le plus d'expérience.

### 6.1 Composants du modèle

Dans le modèle de réseau du langage de traitement d'appel, un réseau de téléphonie Internet contient deux types de composants.

### 6.1.1 Systèmes d'extrémité

Les systèmes d'extrémité qui génèrent et/ou reçoivent des informations de signalisation et des supports. Cela inclut des appareils téléphoniques simples et complexes, des clients de téléphonie sur PC, et des systèmes vocaux automatisés. Le CPL fait abstraction des détails des capacités de ces appareils. Un système d'extrémité peut générer un appel ; et il peut accepter, rejeter, ou transmettre les appels entrants. Les détails de ce processus (sonnerie, téléphones multi lignes, et ainsi de suite) n'ont pas d'importance pour le CPL.

Pour les besoins du CPL, les passerelles -- par exemple, les appareils qui connectent les appels entre un réseau de téléphonie IP et le RTPC – sont aussi considérés comme des systèmes d'extrémité. D'autres appareils, comme des mixeurs ou des pare-feu, ne sont pas directement traités par le CPL, et on n'en discutera pas ici.

### 6.1.2 Serveurs de signalisation

Les serveurs de signalisation sont des appareils qui relayent ou contrôlent les informations de signalisation. Dans SIP, ce sont des serveurs mandataires, des serveurs de redirection, ou des registraires ; dans H.323, ce sont des portiers.

Les serveurs de signalisation peuvent effectuer trois types d'actions sur les informations d'établissement d'appel. Ils peuvent :
- la mandater : le transmettre à un ou plusieurs réseaux ou systèmes d'extrémité, et retourner une des réponses reçues.
- la rediriger : retourner une réponse qui informe le système envoyeur d'une adresse différente à laquelle il devrait envoyer la demande.
- la rejeter : informer le système envoyeur que la demande d'établissement n'a pas pu être menée à bien.

La [RFC2543] illustre des fonctions de mandataire et de redirection. Les systèmes d'extrémité peuvent aussi être capables d'effectuer certaines de ces actions : presque toujours le rejet, et parfois la redirection.

Les serveurs de signalisation conservent normalement aussi les informations sur les localisations d'utilisateurs. Que ce soit au moyen d'enregistrements (messages REGISTER de SIP ou RAS de H.323), par configuration statique, ou par des recherches dynamiques, les serveurs de signalisation doivent avoir des moyens pour déterminer où est actuellement situé un usager, afin de faire des choix intelligents sur leur comportement de mandataire ou de redirection.

Les serveurs de signalisation sont aussi habituellement capables de garder des registres des transactions qui passent à travers eux, et d'envoyer des messages électroniques aux destinations qui sont sur l'Internet, sous le contrôle d'un programme.

## 6.2 Interactions des composants

Lorsque un système d'extrémité passe un appel, la demande d'établissement de l'appel peut passer par divers chemins à travers les composants du réseau. Pour commencer, le système d'extrémité générateur doit décider où envoyer ses demandes. Il y a ici deux possibilités : le générateur peut être configuré de telle sorte que toutes ses demandes aillent à un seul serveur local ; ou il peut résoudre l'adresse de destination pour localiser un serveur de signalisation distant ou un système d'extrémité auquel il peut envoyer directement la demande.

Une fois que la demande arrive à un serveur de signalisation, ce serveur utilise sa base de données de localisation d'usager, sa politique locale, la résolution par le DNS, ou d'autres méthodes, pour déterminer le prochain serveur de signalisation ou système d'extrémité auquel la demande devrait être envoyée. Une demande peut passer à travers un nombre quelconque de serveurs de signalisation : de zéro (dans le cas de communication directe des systèmes d'extrémité) à, en principe, chaque serveur du réseau. De plus, tout système d'extrémité ou serveur de signalisation peut (en principe) recevoir des demandes de tous les autres ou les leur envoyer.

Par exemple, dans la Figure 1, il y a deux chemins que peuvent prendre les informations de demande d'établissement d'appel. Pour Route 1, le générateur connaît seulement une adresse d'usager pour l'utilisateur qu'il essaye de contacter, et il est configuré pour envoyer les appels sortants à travers un serveur mandataire sortant local. Donc, il transmet la demande à son serveur local, qui trouve le serveur d'enregistrement pour cette adresse, et la fait suivre sur ce serveur.

Dans ce cas, l'organisation à laquelle appartient la destination utilise un établissement en plusieurs étapes pour trouver les usagers. Le serveur d'entreprise identifie le département dont fait partie un usager, puis transmet la demande au serveur du département approprié, qui en fait localise l'usager. (Ceci est similaire à la façon dont est souvent configurée la transmission de la messagerie électronique.) La réponse à la demande va voyager en retour sur le même chemin.

Cependant, pour Route 2, le générateur connaît l'adresse spécifique de l'appareil qu'il essaye de contacter, et il n'est pas

configuré à utiliser un mandataire local sortant. Dans ce cas, le générateur peut contacter directement la destination sans avoir du tout à communiquer avec un serveur du réseau.

On voit donc que les serveurs de signalisation de la téléphonie Internet ne peuvent en général pas connaître l'état des systèmes d'extrémité qu'ils "contrôlent" car les informations de signalisation peuvent les avoir outrepassés. Cette limitation architecturale implique un certain nombre de restrictions sur la façon dont certains services peuvent être mis en œuvre. Par exemple, un système réseau ne peut pas savoir de façon fiable si un système d'extrémité est actuellement occupé ou non ; un appel peut avoir été passé au système d'extrémité sans traverser ce système réseau. Donc, les messages de signalisation doivent passer explicitement par les systèmes d'extrémité pour découvrir leur état ; dans cet exemple, le système d'extrémité doit explicitement retourner une indication "occupé".

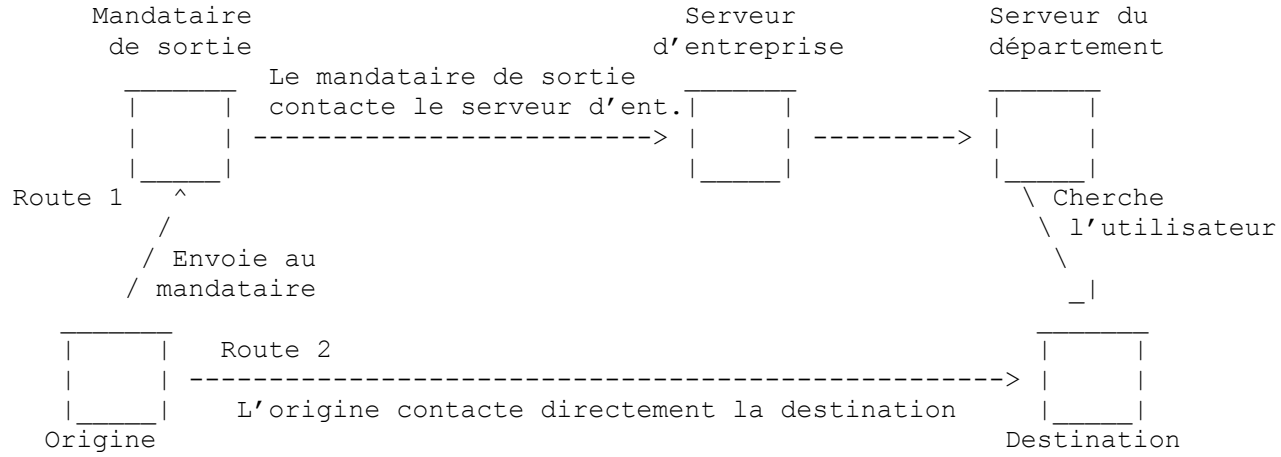


**Figure 1 : Chemins possibles des messages d'établissement d'appel**

## 7. Interaction de CPL avec le modèle de réseau

### 7.1 Ce que fait un script

Un script CPL fonctionne dans un serveur de signalisation, et contrôle les actions de mandatement, de redirection ou de rejet de ce système pour l'établissement d'un appel particulier. Il ne tente pas de coordonner le comportement de plusieurs serveurs de signalisation, ou de décrire des caractéristiques sur un "plan fonctionnel global" comme dans l'architecture de réseau intelligent [Q.1211].

Plus précisément, un script remplace la fonctionnalité de localisation de l'utilisateur d'un serveur de signalisation. Comme décrit au paragraphe 6.1.2, un serveur de signalisation tient normalement une base de données des localisations où un usager peut être joint ; il prend ses décisions de mandatement, de redirection, et de rejet sur la base du contenu de cette base de données. Un script CPL remplace cette fonction de recherche dans la base de données ; il considère les informations d'enregistrement, les spécificités d'une demande d'appel, et les autres informations externes qu'il veut référencer, et choisit les actions de signalisation à effectuer.

De façon abstraite, un script peut être considéré comme une liste de paires de condition/action ; si un attribut de l'enregistrement, de la demande, et des informations externes correspond à une certaine condition, l'action correspondante (ou plus précisément l'ensemble d' actions) est alors entreprise. Dans certaines circonstances, des actions supplémentaires peuvent être entreprises sur la base des conséquences de la première action et de conditions supplémentaires. Si aucune condition ne correspond à l'invitation, c'est l'action standard du serveur de signalisation – par exemple, sa recherche dans la base de données de localisation – qui est entreprise.

### 7.2 Quel script est exécuté

Les scripts CPL sont normalement associés à une certaine adresse de téléphonie Internet. Lorsque arrive une demande d'établissement d'appel à un serveur de signalisation qui est un serveur CPL, ce serveur associe les adresses de source et de destination spécifiées dans la demande à sa base de données de scripts CPL ; si il en est une qui correspond, le script correspondant est exécuté.

Une fois que le script s'est exécuté, si il a choisi d'effectuer une action de mandatement, une nouvelle adresse de téléphonie

Internet va résulter comme destination de ce mandatement. Quand c'est fait, le serveur va à nouveau vérifier dans sa base de données de scripts pour voir si l'un d'entre eux est associé à la nouvelle adresse ; si c'est le cas, ce script sera aussi exécuté ( en supposant qu'un script n'a pas tenté de se mandater sur une adresse que le serveur a déjà essayée). Pour plus de détails sur ce processus de récurrence, et pour une description de ce qui arrive lorsque un serveur a des scripts qui correspondent à la fois à l'adresse d'origine d'un script et à son adresse de destination, voir au paragraphe 9.2.

En général, dans un réseau de téléphonie Internet, une adresse va noter une des deux choses suivantes : un usager, ou un appareil. L'adresse d'un usager se réfère à un individu particulier, par exemple, sip:joe@example.com, sans considérer où est en réalité l'usager ou quelle sorte d'appareil il utilise. À l'opposé, une adresse d'appareil, se réfère à un appareil physique particulier, comme dans sip:x26063@phones.example.com. D'autres sortes d'adresses, intermédiaires, sont aussi possibles, et ont aussi leur utilité (comme une adresse pour "mon téléphone cellulaire, où qu'il soit enregistré actuellement") mais on s'attend à ce qu'elles soient moins courantes. Un script CPL est indifférent au type d'adresse auquel il est associé ; alors que les scripts associés à des adresses d'usagers sont probablement les plus utiles pour la plupart des services, il n'y a pas de raison qu'un script ne puisse pas être associé aussi à tout autre type d'adresse. Le processus de récurrence décrit ci-dessus permet aux scripts d'être associés à plusieurs adresses d'usagers ; et donc, un script d'usager pourrait spécifier une action "essaye à mon téléphone cellulaire", tandis qu'un script d'appareil pourrait dire "je ne veux pas accepter d'appel sur mon téléphone cellulaire quand je suis hors de ma zone de rattachement".

Il est aussi possible à un script CPL d'être associé non pas à une adresse de téléphonie Internet spécifique, mais plutôt à toutes les adresses traitées par un serveur de signalisation, ou un grand ensemble d'entre elles. Par exemple, un administrateur pourrait configurer un système à empêcher les appels provenant d'une liste d'adresses entrantes ou sortantes interdites ; elle devrait vraisemblablement être configurée pour tout le monde, mais les usagers devraient quand même être capables d'avoir aussi leurs propres scripts personnalisés. Le moment exact du processus de récurrence où de tels scripts devraient être exécutés dépend de la nature précise du script administratif. Voir un exposé plus détaillé de cette question au paragraphe 9.2.

### 7.3 Où fonctionne un script

Les usagers peuvent avoir des scripts CPL sur tout serveur réseau par lequel passent leurs demandes d'établissement d'appel et avec lequel ils ont une relation de confiance. Par exemple, dans le cas de la Figure 1, l'usager générateur pourrait avoir un script sur le mandataire de sortie, et l'usager de destination pourrait avoir des scripts à la fois sur le serveur d'entreprise et sur le serveur du département. Ces scripts effectueraient normalement des fonctions différentes, en rapport avec le rôle du serveur sur lequel ils résident ; un script sur le serveur d'entreprise pourrait être utilisé pour personnaliser, par exemple, le département où l'usager souhaite être trouvé, tandis qu'un script au serveur du département pourrait être utilisé pour une personnalisation plus raffinée de la localisation. Certains services, comme de filtrer les appels indésirables, pourraient être situés sur l'un ou l'autre serveur. Voir au paragraphe 9.3 certaines implications d'un tel scénario.

Ce modèle ne spécifie pas les moyens par lesquels les usagers localisent un serveur réseau à capacité CPL. En général, ce sera par les mêmes moyens que ceux par lesquels ils localisent un serveur de téléphonie Internet local auprès duquel s'enregistrer ; cela peut se faire par configuration manuelle, ou par des moyens automatiques tels que le protocole de localisation de service (SLP, *Service Location Protocol)* [RFC2608]. Il a été proposé que les moyens automatisés de localisation de tels serveurs devraient inclure un champ pour indiquer si le serveur permet aux usagers de télécharger des CPL.

## 8. Création et transport d’un script en langage de traitement d’appel

Les usagers créent des scripts en langage de traitement d'appel, normalement sur les appareils d'extrémité, et les transmettent à travers le réseau aux serveurs de signalisation. Les scripts persistent dans les serveurs de signalisation jusqu'à ce qu'ils soient changés ou supprimés, sauf si on leur a spécifiquement donné une heure d'expiration ; un système réseau qui prend en charge les scripts CPL aura besoin d'une mémorisation stable.

L'appareil d'extrémité sur lequel l'usager crée le script CPL n'a pas besoin d'entretenir de relation avec les appareils d'extrémité auxquels sont réellement passés les appels. Par exemple, un script CPL pourrait être créé sur un PC, tandis que les appels pourraient être destinés à être reçus sur un simple téléphonique uniquement audio. Bien sûr, l'appareil sur lequel le script est créé ne peut pas du tout être un "appareil d'extrémité" au sens décrit au paragraphe 6.1.1 ; par exemple, un usager pourrait créer et télécharger un script CPL à partir d'un terminal de la Toile sans capacité multimédia.

Le CPL n'a pas non plus nécessairement besoin d'être créé sur un appareil proche de l'appareil d'extrémité ni du serveur de signalisation en termes de réseau. Par exemple, un usager peut décider de transmettre son appel à un site distant seulement après être arrivé à ce site.

Les moyens exacts par lesquels l'appareil d'extrémité transmet le script au serveur restent à déterminer ; il est vraisemblable que de nombreuses solutions pourront coexister. Cette méthode devra être authentifiée dans presque tous les cas. Les méthodes qui ont été suggérées incluent le téléchargement de fichier de la Toile, les charges utiles de message SIP REGISTER, l'invocation de méthode à distance, SNMP, ACAP, LDAP, et des systèmes de fichier distant comme NFS.

Les usagers peuvent aussi récupérer leur script actuel du réseau sur un système d'extrémité afin qu'il soit édité. Le serveur de signalisation devrait aussi être capable de faire rapport à l'"usager des erreurs relatives au script, aussi bien des erreurs statiques qui pourraient être détectées au moment du chargement que des erreurs qui surviennent pendant le fonctionnement.

Si un usager a une relation de confiance avec plusieurs serveurs de signalisation (comme exposé au paragraphe 7.3) il peut choisir de télécharger des scripts sur tout ou partie de ces serveurs. Ces scripts peuvent être complètement indépendants.

## 9. Comportement d'interaction de caractéristique

Interaction de caractéristique est le terme utilisé dans les systèmes de téléphonie lorsque deux caractéristiques demandées ou plus produisent des comportements ambigus ou conflictuels [IOS Press]. Les questions d'interaction de caractéristiques pour les caractéristiques mises en œuvre avec un langage de traitement d'appel peuvent en gros être divisées en trois catégories : de caractéristique à caractéristique dans un serveur, de script à script dans un serveur, et de serveur à serveur.

### 9.1 Interactions de caractéristique à caractéristique

Du fait de la nature explicite des événements discutés au paragraphe précédent, l'interaction de caractéristique à caractéristique ne va vraisemblablement pas poser de problème dans un environnement de langage de traitement d'appel. Tandis qu'un abonné aux dispositifs de téléphone traditionnel peut sans y penser s'abonner à la fois à "l'appel en attente" et au "renvoi d'appel sur occupation", un usager qui crée un script CPL serait seulement capable de déclencher une action en réponse à la condition "un appel arrive alors que la ligne est occupée". Avec une bonne interface d'usager pour sa création, ou avec un serveur CPL qui peut vérifier le code d'inaccessibilité dans un script téléchargé, les paires de conditions/actions contradictoires peuvent être évitées.

### 9.2 Interaction de script à script

Des interactions de script à script surviennent lorsque un serveur invoque plusieurs scripts pour un seul appel, comme décrit au paragraphe 7.2. Cela peut se produire dans un certain nombre de cas : si l'origine et la destination de l'appel ont toutes deux des scripts spécifiés sur un seul serveur ; si un script fait suivre une demande à une autre adresse qui a aussi un script ; ou si un script administratif est spécifié concurremment à un script individuel d'un usager.

La solution à cette interaction est de déterminer un ordre parmi les scripts à exécuter. Dans cet ordre, le "premier" script est exécuté en premier ; si ce script admet ou permet que l'appel soit mandaté, le script correspondant à la prochaine adresse est exécuté. Lorsque le premier script dit de faire suivre la demande à quelque autre adresse, cette action est considérée comme une nouvelle demande qui arrive au second script. Quand le second script renvoie une réponse finale, cette réponse arrive au premier script de la même manière que si une demande arrivait sur le réseau. Noter que dans certains cas, la transmission peut être récurrente ; un serveur CPL doit veiller à empêcher les boucles de transmission.

De façon abstraite, on peut voir cela comme équivalent à avoir chacun des scripts exécuté sur un serveur de signalisation séparé. Comme l'architecture de CPL est conçue pour permettre que les scripts soient exécutés sur plusieurs serveurs de signalisation dans le cours de la localisation d'un usager, on peut concevoir de transformer les interactions de script à script en interactions de serveur à serveur, décrites au paragraphe suivant, réduisant le nombre de types d'interactions dont on a à se soucier.

La question est alors de déterminer l'ordre correct des scripts. Pour le cas de la transmission d'un script à une adresse qui a aussi un script, l'ordre est évident ; les deux autres cas sont un peu plus subtils. Lorsque il existe à la fois un script d'origine et de destination, le script d'origine devrait être exécuté avant celui de la destination ; cela permet à l'origine d'effectuer la traduction d'adresse, le filtrage d'appel, etc., avant que soit déterminée une adresse de destination et que soit choisi un script correspondant.

Encore plus compliqué est le cas de l'ordre des scripts administratifs. De nombreux scripts administratifs, tels que ceux qui

interdisent les adresses de source et de destination, ne doivent fonctionner qu'après les scripts de l'origine, mais avant les scripts de destination, pour éviter qu'un script de l'usager contourne les interdictions administratives par une transmission habile ; cependant, d'autres fonctions, telles que la traduction d'un annuaire mondial, devront fonctionner tôt ou tard. Les serveurs qui permettent que fonctionnent les scripts administratifs auront besoin de permettre à l'administrateur de configurer le moment où doit se produire le processus d'exécution d'un script administratif particulier.

### 9.3 Interactions de serveur à serveur

Le troisième cas d'interactions de caractéristiques, les interactions de serveur à serveur, est le plus complexe des trois. L'exemple canonique de ce type d'interaction est la combinaison du filtrage des appels sortants et du renvoi d'appel : un usager (ou administrateur) peut souhaiter empêcher de passer des appels à une certaine adresse, mais le script local n'a aucun moyen de savoir si un appel passé à une autre adresse, légitime, ne va pas être mandaté par un serveur distant, à l'adresse défendue. Ce type de problème est insoluble dans un réseau administrativement hétérogène, même un réseau "légèrement" hétérogène comme un système de téléphonie actuel. Le CPL ne prétend pas résoudre ce problème mais il n'est pas pire pour les scripts de CPL que pour n'importe quel autre moyen de déployer des services.

Une autre classe d'interactions de serveur à serveur est mieux résolue par le protocole de signalisation sous-jacent, car elles peuvent se produire que les serveurs de signalisation soient contrôlés par un langage de traitement d'appel ou par de tout autres moyens. Un exemple en est la boucle de transmission, où un usager X peut avoir des appels transmis à Y, qui a ses appels retransmis à X. SIP a un mécanisme pour détecter de telles boucles. Un serveur de langage de traitement d'appel n'a donc pas besoin de définir de mécanisme spécial pour empêcher que cela se produise ; il devrait cependant être possible de déclencher un ensemble différent d'actions de traitement de l'appel dans le cas de détection d'une boucle, et/ou de faire rapport d'une erreur au propriétaire du script par un mécanisme normalisé de rapport d'erreur en cours de fonctionnement.

### 9.4 Signalisation ambiguë

De plus, [IOS Press] expose un quatrième type d'interaction de caractéristiques pour les réseaux téléphoniques traditionnels, les ambiguïtés de signalisation. Cela peut se produire lorsque plusieurs caractéristiques surchargent la même opération dans le canal sémaphore limité d'une station d'extrémité au réseau : par exemple, faire clignoter le signal de connexion peut signifier aussi bien "ajouter un participant à un appel à trois" que "mise en attente de l'appel". À cause de la nature explicite de la signalisation dans les protocoles de téléphonie Internet discutés ici, cette question ne se pose pas.

## 10. Relations avec les langages existants

La description par le présent document de CPL comme un "langage" n'est pas destinée à impliquer qu'un nouveau langage doit nécessairement être mis en œuvre à partir de rien. Un serveur pourrait éventuellement mettre en œuvre toutes les fonctionnalités décrites ici comme une bibliothèque ou un ensemble d'extensions pour un langage existant ; Java, ou les divers langages librement accessibles de script (Tcl, Perl, Python, Guile) sont des possibilités évidentes.

Cependant, il y a des raisons pour créer un nouveau langage. Tous les langages existants sont, naturellement, expressivement complets ; cela présente deux inconvénients inhérents. Le premier est que toute fonction mise en œuvre en eux peut prendre un temps arbitrairement long, utiliser une quantité de mémoire arbitrairement grande, et peut ne jamais se terminer. Pour le traitement d'appel, cette sorte d'utilisation des ressource n'est probablement pas nécessaire, et, comme décrit au paragraphe 12.1, peut en fait être indésirable. Le langage de filtrage de messagerie électronique Sieve [RFC3028], qui s'interdit délibérément d'être complet au sens de Turing, en est un modèle.

On pourrait obtenir des niveaux de sécurité et protection similaires (bien que leur génération et leur analyse ne soient pas automatiques) par l'usage d'une "boîte à sable" comme celle qui est utilisée dans les appliquettes Java, où des limites strictes sont imposées à la quantité de mémoire, de temps de cpu, d'espace de pile, etc., que peut utiliser un programme. La difficulté avec cette approche est principalement dans son manque de transparence et de portabilité : sauf si les niveaux de ces limites sont imposés par la norme, c'est une mauvaise idée tant que les ressources disponibles augmentent de façon exponentielle avec la loi de Moore, un usager ne peut jamais être sûr qu'un certain programme peut réussir à être exécuté sur un certain serveur sans atteindre les limites de ressource du serveur, et un programme qui s'exécute bien sur un serveur peut échouer de façon inattendue sur un autre. D'un autre côté, les langages non expressivement complets permettent un contact implicite entre l'auteur du script et le serveur : tant que le script reste dans les règles du langage, le serveur va garantir qu'il exécutera le script.

Le second inconvénient des langages expressivement complets est qu'ils rendent la génération et analyse automatiques des

scripts très difficiles, car chaque outil d'analyse doit être un interprète du langage entier. On peut faire une analogie avec le monde de la création de document : alors que les langages de balisage de texte comme HTML ou XML peuvent être, et sont, facilement manipulés par des éditeurs intelligents, des langages de programmation de document puissants tels que LaTeX ou Postscript ne peuvent habituellement pas l'être. Bien qu'il y ait des traitements de texte qui peuvent sauvegarder leurs documents en forme LaTeX, ils ne peuvent pas accepter en entrée des documents LaTeX arbitraires et conserver la structure du document d'origine sous une forme éditée. À l'opposé, pratiquement tout éditeur HTML peut éditer tout document HTML à partir de la Toile, et ceux de haute qualité préservent la structure des documents originaux au cours de leur édition.

## 11. Travaux connexes

### 11.1 Environnements de création de service de réseau intelligent

La série des Recommandations de l'UIT sur le réseau intelligent décrit, à un niveau abstrait, les environnements de création de service [Q.1211]. Elles décrivent les services dans un réseau traditionnel de téléphonie à commutation de circuits comme une série de décisions et d'actions arrangées dans un graphe acyclique dirigé. De nombreux fabricants de services de réseau intelligent utilisent des versions modifiées et étendues de cela pour leurs propres environnements de création de services.

### 11.2 CGI SIP

SIP CGI [9] est une interface pour mettre en œuvre des services sur des serveurs SIP. À la différence d'un CPL, c'est une interface de niveau très inférieur, et ne serait pas appropriée pour des services écrits par des usagers qui ne sont pas de confiance.

L'article "Programming Internet Telephony Services" *(programmer des services de téléphonie Internet)* [CUCS-010-99] expose plus en détails les similitudes et les contrastes entre SIP CGI et CPL.

## 12. Caractéristiques de langage nécessaires

Cette section énumère les propriétés d'un langage de traitement d'appel que nous pensons nécessaires afin de mettre en œuvre les exemples qui en expliquent les motifs, en ligne avec l'architecture décrite.

### 12.1 Caractéristiques de langage

Voici quelques attributs abstraits que toute proposition de langage de traitement d'appel devrait posséder.

o Léger, efficace, facile à mettre en œuvre
En plus des raisons générales de le préférer, un serveur réseau peut théoriquement traiter de très gros volumes d'appels, et on ne veut pas que l'exécution du CPL constitue un goulet d'étranglement majeur. Une façon de le faire pourrait être de compiler les scripts avant exécution.

o Il est facile de vérifier qu'il est correct
Pour un script qui fonctionne sur un serveur, de mauvaises configurations peuvent résulter en ce qu'un usager devienne injoignable, rendant difficile de lui indiquer les erreurs durant le fonctionnement (bien qu'un mécanisme de rapport sur un second canal comme la messagerie électronique puisse améliorer cela). Donc, il devrait être possible de vérifier, lorsque le script est affecté au serveur, qu'il est au moins syntaxiquement correct, ne contient pas de boucle évidente ou autres modes d'échec, et n'utilise pas trop de ressources du serveur.

o Exécutable de manière sûre
Aucune des actions qu'entreprend le script CPL ne devrait être capable de subvertir quelque chose du serveur auquel l'usager ne devrait pas avoir accès, ou affecter sans permission l'état des autres usagers. De plus, comme les scripts CPL vont normalement fonctionner sur un serveur sur lesquels les usagers ne peuvent normalement pas faire fonctionner de code, le langage ou son environnement d'exécution doivent être conçus de telle sorte que les scripts ne puissent pas utiliser des quantités illimitées des ressources du réseau, du temps de CPU du serveur, ou de mémorisation.

o Facile à écrire et analyser par 'homme comme par la machine
Pour un maximum de souplesse, on veut permettre aux personnes d'écrire leurs propres scripts, ou d'utiliser et personnaliser

des bibliothèques de scripts fournies par d'autres. Cependant, la plupart des utilisateurs voudront avoir une interface d'usager plus intuitive pour les mêmes fonctionnalités, et auront donc un programme qui créera les scripts pour eux. Les deux cas devraient être faciles ; en particulier, il devrait être facile à un éditeur de scripts de lire les scripts générés par des personnes et vice versa.

o Extensible

Il devrait être possible d'ajouter des caractéristiques supplémentaires à un langage de telle façon que les scripts existants continuent de fonctionner, et que les serveurs existants puissent facilement reconnaître les caractéristiques qu'ils ne comprennent pas et informer l'"utilisateur de ce fait en toute sécurité.

o Indépendance à l'égard des détails de la signalisation sous-jacente

Les mêmes scripts devraient être utilisables que le protocole sous-jacent soit SIP, H.323, un réseau téléphonique traditionnel, ou tout autre moyen d'établir des appels. Il devrait aussi être indépendant des formats d'adresse. (On utilise la terminologie de SIP dans notre description des exigences, mais cela devrait bien se transposer dans d'autres systèmes.) Il peut aussi être utile d'avoir des extensions du langage pour le traitement d'autres sortes de communication, comme la messagerie électronique ou la télécopie.

### 12.2 Caractéristiques de base – signalisation d'appel

Pour être utile, un langage de traitement d'appel devrait évidement être capable de réagir à, et d'initier, des événements de signalisation d'appel.

o Devrait exécuter les actions quand arrive une demande d'appel. Voir à la Section 7, en particulier au 7.1.

o Devrait être capable de prendre des décisions sur la base des propriétés d'événement

Un certain nombre de propriétés d'un événement d'appel sont pertinentes pour un processus de décision d'un script. Cela inclut, en gros par ordre d'importance :

adresse de destination : on veut être capable de faire un acheminement ou un examen sur la base de la destination. Noter que dans SIP, on veut être capable de filtrer sur l'une ou/et l'autre des adresses dans l'en-tête To et l'URI de demande.

adresse d'origine : de même, on veut être capable de faire un examen ou acheminement fondé sur l'origine.

préférences de l'appelant : dans SIP, un appelant peut exprimer des préférences quant au type d'appareil à atteindre – voir les [RFC3840], [RFC3841]. Le script devrait être capable de prendre des décisions sur la base de ces informations.

informations sur l'appelant ou sur l'appel : SIP a des champs textuels tels que Subject, Organization, Priority, etc., et un nom d'affichage pour les adresses ; les usagers peuvent aussi ajouter des en-têtes supplémentaires non standard. H.323 a un seul champ Display *(Affichage)*. Le script devrait être capable de prendre des décisions sur la base de ces paramètres.

description du support : les invitations d'appel spécifient les types de support sur lesquels elles vont s'écouler, l'utilisation de leur bande passante, leurs adresses de destination réseau, etc. Le script devrait être capable de prendre des décisions sur la base de ces caractéristiques de support.

état d'authentification/chiffrement : les invitations d'appel peuvent être authentifiées. De nombreuses propriétés de l'authentification sont pertinentes : la méthode d'authentification/chiffrement, qui a effectué l'authentification, quels champs spécifiques ont été chiffrés, etc. Le script devrait être capable de prendre des décisions sur la base de ces paramètres de sécurité.

o Devrait être capable d'entreprendre une action sur la base d'une invitation d'appel

Un certain nombre d'actions peuvent être entreprises en réponse à une demande d'établissement d'appel entrant. On peut :

le rejeter : on devrait être capable d'indiquer que l'appel n'est pas acceptable ou qu'on est pas capable de le mener à bien. On devrait aussi être capable d'envoyer des codes de rejet plus spécifiques (y compris, pour SIP, la chaîne textuelle associée, les codes d'annonce, ou la charge utile de message).

le rediriger : on devrait être capable de dire à l'envoyeur qui a initié l'appel d'essayer à un site différent.

le mandater : on devrait être capable d'envoyer l'invitation d'appel sur une autre localisation, ou sur plusieurs autres localisations ("fourcher" l'invitation), et attendre les réponses. Il devrait aussi être possible de spécifier une valeur de temporisation après laquelle on abandonne la réception des réponses.

o Devrait être capable d'entreprendre une action sur la base d'une réponse à une invitation d'appel mandatée ou fourchée.

Une fois qu'on a mandaté une invitation, on a besoin d'être capable de prendre des décisions sur la base des réponses qu'on reçoit à cette invitation (ou de son absence). On devrait être capable de :

- considérer ses champs de message : on devrait être capable de considérer les mêmes champs dans une réponse que dans l'invitation initiale.
- la relayer à l'origine de l'appel : si la réponse est satisfaisante, elle devrait être retournée à l'envoyeur.
- pour une fourche, choisir une des réponses à relayer : si une invitation est fourchée, on s'attend évidemment à recevoir plusieurs réponses. Cela pose plusieurs problèmes – de choisir parmi les réponses, et pendant combien de temps

attendre si on n'a pas reçu de réponse de toutes les destinations.
- initier d'autres actions : si on n'a pas obtenu de réponse, ou pas celle qu'on voudrait, on devrait être capable d'essayer autre chose à la place (par exemple, transfert d'appel sur occupation).

### 12.3 Caractéristiques de base – non signalisation

Un certain nombre d'autres caractéristiques que devraient avoir un langage de traitement d'appel ne se réfèrent pas en elles-mêmes à la signalisation d'appel ; cependant, il est quand même très souhaitable qu'il mette en œuvre de nombreuses caractéristiques utiles.

Les serveurs qui fournissent ces caractéristiques peuvent résider dans d'autres appareils de l'Internet, ou pourraient être localisés dans le serveur (ou autres possibilités). Le langage devrait être indépendant de la localisation de ces serveurs, au moins à un haut niveau.

o Journalisation
En plus de l'enregistrement naturel des événements du serveur de CPL, l'usager va aussi vouloir être capable d'enregistrer d'autres éléments arbitraires. La mémorisation réelle de ces informations de journalisation peut se faire en local ou à distance.

o Rapport d'erreurs
Si une erreur inattendue survient, le script devrait être capable de faire rapport de l'erreur au propriétaire du script. Cela peut utiliser le même mécanisme que celui qu'utilise le serveur de script pour faire rapport des erreurs de langage à l'utilisateur (voir au paragraphe 12.5).

o Accès aux informations de localisation de l'utilisateur
Les mandataires vont souvent collecter les informations sur la localisation actuelle de l'utilisateur, soit par des messages SIP REGISTER, la famille de messages RAS RRQ de H.323, soit par quelque autre mécanisme (voir au paragraphe 6.2). Le CPL devrait être capable de se référer à ces informations afin qu'un appel puisse être transmis aux localisations enregistrées ou à un sous-ensemble d'entre elles.

o Accès aux bases de données
Beaucoup des informations du contrôle du CPL pourraient être mémorisées dans des bases de données externes, par exemple dans une base de données d'adresse de grande zone, ou des informations d'autorisation, pour un CPL sous contrôle administratif. Le langage pourrait spécifier des protocoles d'accès spécifiques de la base de données (comme SQL ou LDAP) ou pourraient être plus génériques.

o Autres informations externes
D'autres informations externes auxquelles un script pourrait accéder incluent des pages de la Toile, qui pourraient être renvoyées dans un corps de message SIP ; ou une interface propre à des appels de procédure à distance tels que Corba, RMI, ou DCOM, par exemple pour accéder à une base de données externe de facturation. Cependant, pour simplifier, ces interfaces pourront n'être pas dans la version initiale du protocole.

### 12.4 Caractéristiques de langage

Certaines caractéristiques n'impliquent aucune opération externe à l'environnement d'exécution du CPL, mais sont quand même nécessaires pour permettre de mettre en œuvre certains services standard. (Cette liste n'est pas exhaustive.)

o Conformité à un gabarit
Il devrait être possible de réserver un traitement particulier aux adresses et autres chaînes textuelles sur la base non seulement de la chaîne complète, mais aussi sur des sous-schémas plus généraux ou complexes.

o Filtrage d'adresse
Une fois qu'un ensemble d'adresses a été restitué par une des méthodes du paragraphe 12.3, l'usager n'a pas besoin d'être capable de choisir un sous-ensemble de celles-ci, sur la base de leurs composants d'adresse ou d'autres paramètres.

o Aléation
Certaines formes de distribution d'appel sont rendues aléatoires quant à la façon dont elles se terminent en pratique.

o Informations de date et d'heure
Les utilisateurs peuvent souhaiter conditionner certains services (par exemple, le renvoi d'appel, la distribution des appels)

à l'heure en cours, ou au jour de la semaine, etc.

### 12.5 Contrôle

Comme décrit à la Section 8, on doit avoir un mécanisme pour envoyer et restituer les scripts CPL, et les données associées, de et vers un serveur de signalisation. Cette méthode devrait prendre en charge le rapport aux utilisateurs des erreurs au moment du chargement ; on a aussi besoin d'un mécanisme pour faire rapport des erreurs au moment de l'exécution du script. L'authentification est vitale, et le chiffrement est très utile. La spécification de ce mécanisme peut être (et devrait probablement être) une spécification distincte de celle du langage de traitement d'appel lui-même.

## 13. Considérations pour la sécurité

Les considérations pour la sécurité du transfert des scripts CPL sont discutés à la section 8 et au paragraphe 12.5. Des considérations sur l'exécution du langage sont exposées au paragraphe 12.1.

## 14. Remerciements

Nous tenons à remercier Tom La Porta et Jonathan Rosenberg de leurs commentaires et suggestions.

## 15. Adresse des auteurs

Jonathan Lennox
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
mél : lennox@cs.columbia.edu

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
mél : schulzrinne@cs.columbia.edu

## 16. Bibliographie

[CUCS-010-99] J. Rosenberg, J. Lennox, and H. Schulzrinne, "Programming internet telephony services," Technical Report CUCS-010-99, Columbia University, New York, mars 1999.

[H.323] Recommandation UIT-T H.323, "Systèmes de communication multimédia fondées sur le paquet", Genève, Confédération Helvétique, février 1998.

[IOS Press] E. J. Cameron, N. D. Griffeth, Y.-J. Lin, M. E. Nilson, W. K. Schure, and H. Velthuijsen, "A feature interaction benchmark for IN and beyond," Feature Interactions in Telecommunications Systems, IOS Press, pp. 1-23, 1994.

[Q.1211] Recommandation UIT-T Q.1211, "Recommandations générales sur la commutation et la signalisation téléphonique – Réseau intelligent : Introduction aux capacités du réseau intelligent, ensemble 1", Genève, Confédération Helvétique, mars 1993.

[RFC2543] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, "SIP : protocole d'initialisation de session", mars 1999. (*Obsolète, voir* RFC3261, RFC3262, RFC3263, RFC3264, RFC3265) *(P.S.)*

[RFC2608] E. Guttman et autres, "Protocole de localisation de service, version 2", juin 1999. (*MàJ par* RFC3224) *(P.S.)*

[RFC3028] T. Showalter, "Sieve : Langage de filtrage de messagerie", janvier 2001. *(Obsolète, voir la* RFC5228*) (P.S.)*

[RFC3050] J. Lennox, H. Schulzrinne, J. Rosenberg, "Interface commune de routeur pour SIP", janvier 2001. *(Info.)*

[RFC3840] J. Rosenberg, H. Schulzrinne et P. Kyzivat, "Indication des capacités d'agent d'utilisateur dans le protocole

d'initialisation de session (SIP)", août 2004

[RFC3841] J. Rosenberg, H. Schulzrinne, P. Kyzivat, "Préférences de l'appelant pour le protocole d'initialisation de session (SIP)", août 2004. *(P.S.)*

[RFC3875] D. Robinson, K. Coar, "Interface de passerelle commune (CGI) version 1.1", octobre 2004. *(Information)*

[RFC3880] J. Lennox, X. Wu, H. Schulzrinne, "Langage de traitement d'appel (CPL) : un langage pour le contrôle d'usager des services de téléphonie Internet", octobre 2004.

## 17. Déclaration complète de droits de reproduction



## Remerciement

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