Groupe de travail Réseau
Request for Comments : 5629
Catégorie : Sur la voie de la normalisation

J. Rosenberg, Cisco Systems
octobre 2009
Traduction Claude Brière de l'Isle

Cadre d'interaction d'applications dans le protocole d'initialisation de session (SIP)

Résumé
Le présent document décrit un cadre pour l'interaction entre les utilisateurs et les applications fondées sur le protocole d'initialisation de session (SIP, *Session Initiation Protocol*). En interagissant avec les applications, les utilisateurs peuvent guider la façon dont ils opèrent. Le but de ce cadre est la signalisation de stimulus, qui permet à un agent d'utilisateur (UA) d'interagir avec une application sans connaissance de la sémantique de cette application. La signalisation de stimulus peut se produire à une interface d'utilisateur fonctionnant en local avec le client, ou à une interface d'utilisateur distante, à travers un flux de supports. La signalisation de stimulus englobe une large gamme de mécanismes, allant de cliquer sur un hyperlien, de presser des boutons, à l'entrée traditionnelle de multifréquence bi tonalités (DTMF, *Dual-Tone Multi-Frequency*). Dans tous les cas, la signalisation de stimulus est prise en charge par l'utilisation de langages de balisage, qui jouent un rôle clé dans ce cadre.

Statut de ce mémoire
Ceci est un document de l'Internet sur la voie de la normalisation.

Le présent document a été produit par l'équipe d'ingénierie de l'Internet (IETF). Il représente le consensus de la communauté de l'IETF. Il a subi une révision publique et sa publication a été approuvée par le groupe de pilotage de l'ingénierie de l'Internet (IESG). Tous les documents approuvés par l'IESG ne sont pas candidats à devenir une norme de l'Internet ; voir la Section 2 de la RFC5741.

Les informations sur le statut actuel du présent document, tout errata, et comment fournir des réactions sur lui peuvent être obtenues à http://www.rfc-editor.org/info/rfc5629

Table des matières

1. Introduction

Le protocole d'initialisation de session (SIP, *Session Initiation Protocol*) [RFC3261] fournit aux utilisateurs la capacité d'initier, gérer, et terminer des sessions de communications. Fréquemment, ces sessions vont impliquer une application SIP. Une application SIP est définie comme un programme fonctionnant sur un élément fondé sur SIP (comme un mandataire ou un agent d'utilisateur) qui fournit une fonction à valeur ajoutée à un utilisateur ou administrateur de système. Des exemples d'applications SIP incluent des appels sur carte pré-payée, des conférences, et l'acheminement d'appels fondés sur la présence [RFC2778].

Afin que la plupart des applications fonctionnent correctement, elles ont besoin d'entrées de la part de l'utilisateur pour guider leur fonctionnement. Par exemple, une application de carte d'appel prépayée exige que l'utilisateur entre son numéro de carte d'appel, son code PIN, et le numéro de destination qu'il souhaite joindre. Le processus par lequel un utilisateur fournit des entrées à une application est appelé une "interaction d'application".

L'interaction d'application peut être fonctionnelle ou par stimulus. L'interaction fonctionnelle exige que l'appareil d'utilisateur comprenne la sémantique de l'application, tandis que l'interaction par stimulus ne l'exige pas. La signalisation par stimulus permet que les applications soient construites sans exiger de modifications à l'appareil d'utilisateur. L'interaction par stimulus est l'objet du présent cadre. Le cadre fournit un modèle de la façon dont les utilisateurs interagissent avec les applications à travers les interfaces d'utilisateur, et comment les interfaces et applications d'utilisateur peuvent être distribuées à travers un réseau. Ce modèle est ensuite utilisé pour décrire comment les applications peuvent instancier et gérer les interfaces d'utilisateur.

2. Conventions utilisées dans ce document

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].

3. Définitions

Application SIP : une application SIP est définie comme un programme fonctionnant sur un élément fondé sur SIP (comme

un mandataire ou un agent d'utilisateur) qui fournit une fonction à valeur ajoutée à un utilisateur ou administrateur de système. Des exemples d'applications SIP incluent des appels par carte prépayée, des conférences, et des acheminements d'appel fondés sur la présence [RFC2778].

Interaction d'application : processus par lequel un utilisateur fournit des entrées à une application.

Interaction d'application en temps réel : interaction d'application qui a lieu pendant qu'une instance d'application s'exécute. Par exemple, quand un utilisateur entre son numéro de PIN dans une application de carte d'appel prépayée, ce qui est une interaction d'application en temps réel .

Interaction d'application non en temps réel : interaction d'application qui a lieu sans rapport avec l'exécution de l'application. Généralement, une interaction d'application non en temps réel est accomplie par provisionnement.

Interaction d'application fonctionnelle : une interaction d'application est fonctionnelle quand l'appareil de l'utilisateur comprend la sémantique de l'interaction avec l'application.

Interaction d'application de stimulus : une interaction d'application est de stimulus quand l'appareil de l'utilisateur ne comprend pas la sémantique de l'interaction avec l'application.

Interface d'utilisateur (UI, *User Interface*) : l'interface d'utilisateur fournit à l'utilisateur le contexte pour prendre des décisions sur ce qu'il veut. L'utilisateur interagit avec l'appareil, qui transporte l'entrée de l'utilisateur à l'interface d'utilisateur. L'interface d'utilisateur interprète les informations et les passe à l'application.

Composant d'interface d'utilisateur : élément de l'interface d'utilisateur qui opère indépendamment des autres éléments de l'interface d'utilisateur. Par exemple, un utilisateur pourrait avoir deux interfaces séparées de la Toile à une application de carte d'appel prépayée : une pour décrocher et faire un autre appel, et une autre pour entrer le nom d'utilisateur et le PIN.

Appareil d'utilisateur : système logiciel ou matériel avec lequel l'utilisateur interagit directement pour communiquer avec l'application. Un exemple d'appareil d'utilisateur est un téléphone. Un autre exemple est un ordinateur personnel avec un navigateur.

Mandataire d'appareil d'utilisateur : système logiciel ou matériel à travers lequel l'utilisateur interagit indirectement pour communiquer avec l'application. Ce lien indirect peut être à travers un réseau. Un exemple est une passerelle de IP au réseau téléphonique public commuté (RTPC). Elle agit comme un mandataire d'appareil d'utilisateur, agissant au nom de l'utilisateur sur le réseau de circuits.

Entrée d'utilisateur : informations "brutes" passées d'un utilisateur à une interface d'utilisateur. Des exemples d'entrée d'utilisateur incluent un mot prononcé ou a clic sur un hyperlien.

Interface d'utilisateur de client local : interface d'utilisateur co-résidente avec l'appareil d'utilisateur.

Interface d'utilisateur de client distant : interface d'utilisateur qui s'exécute à distance de l'appareil de l'utilisateur. Dans ce cas, une interface normalisée est nécessaire entre l'appareil d'utilisateur et l'interface d'utilisateur. Normalement, c'est fait par des sessions de support : audio, vidéo, ou partage d'application.

Langage de balisage : un langage de balisage décrit un flux logique de présentation des informations à l'utilisateur, de collecte des informations provenant de l'utilisateur, et de transmission de ces informations à une application.

Interaction de supports : moyen de séparer un utilisateur et une interface d'utilisateur en les connectant avec les flux de supports.

Réponse de voix interactive (IVR, *Interactive Voice Response*) : une IVR est un type d'interface d'utilisateur qui permet aux utilisateurs de prononcer les commandes de l'application, et d'entendre les réponses à ces commandes qui invitent à plus d'informations.

Invite et collecte : primitive de base d'une interface d'utilisateur IVR. Une option vocale est présentée à l'utilisateur, et l'utilisateur dit son choix.

Irruption : acte d'entrer des informations dans une interface d'utilisateur IVR avant l'achèvement d'une invite demandant ces informations.

Convergence : un composant d'interface d'utilisateur a convergé quand l'entrée d'utilisateur lui est fournie, par opposition à tous les autres composants d'interface d'utilisateur. Ceci n'est pas à confondre avec le terme "point de concentration"

dans le cadre de conférence SIP, qui se réfère à l'agent d'utilisateur central d'une conférence [RFC4353].

Détermination de convergence : processus par lequel l'appareil d'utilisateur détermine quel composant d'interface d'utilisateur va recevoir l'entrée d'utilisateur.

Appareil sans convergence : appareil d'utilisateur qui n'a pas de capacité d'effectuer la détermination de la convergence. Un exemple d'appareil sans convergence est un téléphone avec un clavier.

UI capable de présentation : interface d'utilisateur qui peut inviter l'utilisateur à faire une entrée, collecter les résultats, et ensuite inviter l'utilisateur avec de nouvelles informations sur la base de ces résultats.

UI sans présentation : interface d'utilisateur qui ne peut pas inviter l'utilisateur avec des informations.

Interaction de caractéristiques : classe de problèmes qui résulte de ce que plusieurs applications ou composants d'application essayent en même temps de fournir des services à un utilisateur.

Interaction de caractéristiques inter-applications : interactions de caractéristiques qui se produisent entre des applications.

DTMF *(Dual-Tone Multi-Frequency) :* multi-fréquences bi tonalités. DTMF se réfère à une classe de tonalités générées par des appareils de téléphonie à commutation de circuits quand l'utilisateur presse une clé sur le clavier. Par suite, DTMF et entrée de clavier sont souvent utilisées de façon synonyme, quand en fait l'un d'eux (DTMF) est simplement un moyen de transporter l'autre (l'entrée de clavier) à l'interface d'utilisateur de client distant (le commutateur, par exemple).

Instance d'application : un seul chemin d'exécution d'une application SIP.

Application génératrice : application SIP qui agit comme client d'agent d'utilisateur (UAC, *User Agent Client*) faisant un appel au nom de l'utilisateur.

Application de terminaison : application SIP qui agit comme serveur d'agent d'utilisateur (UAS, *User Agent Server*) répondant à un appel généré par un utilisateur. Les applications d'IVR sont des applications de terminaison.

Application intermédiaire : application SIP qui n'est ni l'appelant ni l'appelé, mais plutôt un tiers impliqué dans un appel.

4. Modèle d'interaction d'applications

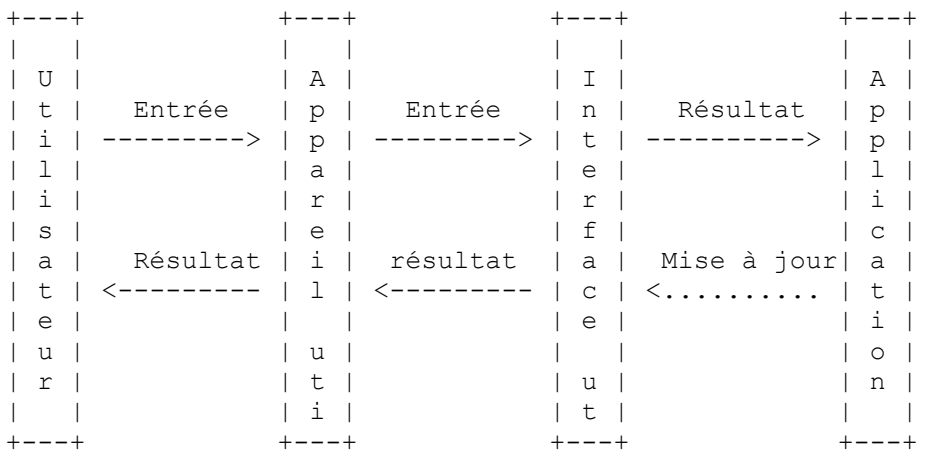


Figure 1 : Modèle d'interactions en temps réel

La Figure 1 présente un modèle général pour la façon dont les utilisateurs interagissent avec les applications. Généralement, les utilisateurs interagissent avec une interface d'utilisateur à travers un appareil d'utilisateur. Un appareil d'utilisateur peut être un téléphone, ou un ordinateur personnel avec un navigateur de la Toile. Son rôle est de passer l'entrée de l'utilisateur de l'utilisateur à l'interface d'utilisateur. L'interface d'utilisateur fournit à l'utilisateur un contexte afin de prendre des décisions sur ce qu'il veut. L'utilisateur interagit avec l'appareil, causant le passage des informations de l'appareil à l'interface d'utilisateur. L'interface d'utilisateur interprète les informations, et les passe à un événement d'interface d'utilisateur à l'application. L'application peut être capable de modifier l'interface d'utilisateur sur la base de cet événement. Si c'est ou non possible dépend du type d'interface d'utilisateur.

Les interfaces d'utilisateur sont fondamentales sur le rendu et l'interprétation. Le rendu se réfère à la façon dont le contexte est fourni à l'utilisateur. Ce peut être par des hyperliens, des images, des sons, des vidéos, du texte, et ainsi de suite. L'interprétation se réfère à la façon dont l'interface d'utilisateur prend les données "brutes" fournies par l'utilisateur, et retourne le résultat à l'application comme un événement significatif, abstraction faite des particularités de l'interface d'utilisateur. Par exemple, considérons une application de carte d'appel prépayée. L'interface d'utilisateur s'inquiète de détails comme quelle invite est fournie à l'utilisateur, si la voix est mâle ou féminine, et ainsi de suite. Elle est concernée par la reconnaissance de parole que fournit l'utilisateur, afin d'obtenir les informations désirées. Dans ce cas, les informations désirées sont le numéro de carte d'appel, le code PIN, et le numéro de destination. L'application a besoin de ces données, et il importe peu à l'application de savoir si elles ont été collectées en utilisant une invite mâle ou non.

Les interfaces d'utilisateur ont généralement des exigences de temps réel à l'égard de l'utilisateur. C'est-à-dire que quand un utilisateur interagit avec l'interface d'utilisateur, celle-ci a besoin de réagir rapidement, et ce changement doit être immédiatement propagé à l'utilisateur. Cependant, l'interface entre l'interface d'utilisateur et l'application n'a pas besoin d'être aussi rapide. Le plus rapide est le mieux, mais l'interface d'utilisateur elle-même peut fréquemment compenser les longues latences entre l'interface d'utilisateur et l'application. Dans le cas de l'application de carte d'appel prépayée, quand l'utilisateur est invité à entrer son PIN, l'invite devrait généralement s'arrêter immédiatement quand le premier chiffre du PIN est entré. C'est ce qui est appelé "irruption" *(barge-in)*. Après que l'interface d'utilisateur a collecté le reste du PIN, elle peut dire à l'utilisateur "prière d'attendre le traitement". Le PIN peut alors être graduellement transmis à l'application. Dans cet exemple, l'interface d'utilisateur a compensé un UI lent à l'interface d'application en demandant à l'utilisateur d'attendre.

La séparation entre interface d'utilisateur et l'application est absolument fondamentale pour tout le cadre fourni dans ce document. Son importance ne peut pas être surestimée.

Avec le modèle de base, on peut commencer la taxonomie des types de systèmes qui peuvent être construits.

4.1 Fonctionnel contre stimulus

La première façon d'établir une taxonomie du système est de considérer l'interface entre l'UI et l'application. Il y a deux modèles fondamentalement différents pour cette interface. Dans une interface fonctionnelle, l'interface d'utilisateur a une connaissance précise de l'application et est, en fait, spécifique de l'application. L'interface entre les deux composants est à travers un protocole fonctionnel, capable de représenter la sémantique qui peut être exposée à travers l'interface d'utilisateur. Parce que l'interface d'utilisateur a connaissance de l'application, elle peut être conçue de façon optimale pour cette application. Par suite, les interfaces fonctionnelles d'utilisateur sont presque toujours les plus conviviales, les plus rapides, et les plus sensibles. Cependant, afin de permettre l'interopérabilité entre appareils utilisateurs et applications, les détails des protocoles fonctionnels doivent être spécifiés dans les normes. Cela ralentit l'innovation et limite la portée des applications qui peuvent être construites.

Une solution de remplacement est une interface de stimulation. Dans une interface de stimulation, l'interface d'utilisateur est générique -- c'est-à-dire, ignorant totalement les détails de l'application. Bien sûr, l'application peut passer des instructions à l'interface d'utilisateur décrivant comment elle devrait opérer. L'interface d'utilisateur traduit l'entrée d'utilisateur en des "stimuli", qui sont des données comprises seulement par l'application, et pas par l'interface d'utilisateur. Parce que elles sont génériques, et parce que elles exigent des communications avec l'application afin de changer la façon de rendre l'information à l'utilisateur, les interfaces d'utilisateur par stimulation sont généralement plus lentes, moins conviviales, et moins sensibles que leur contrepartie fonctionnelle. Cependant, elles permettent une innovation substantielle dans les applications, car aucune activité de normalisation n'est nécessaire pour construire une nouvelle application, tant qu'elle peut interagir avec l'utilisateur dans les limites du mécanisme de l'interface d'utilisateur. La Toile est un exemple d'interface d'utilisateur par stimulus aux applications.

Dans les systèmes SIP, les interfaces fonctionnelles sont fournies en étendant le protocole SIP pour qu'il fournisse la fonction nécessaire. Par exemple, la spécification des préférences de l'appelant SIP [RFC3841] fournit une interface fonctionnelle qui permet à un utilisateur de demander aux applications d'acheminer l'appel à des types d'agents d'utilisateur spécifiques. Les interfaces fonctionnelles sont importantes, mais ne sont pas le sujet du présent cadre. Le but principal du présent cadre est de traiter le rôle des interfaces par stimulation pour les applications SIP.

4.2 Temps réel contre non temps réel

Les systèmes d'interaction d'application peuvent aussi être en temps réel ou non en temps réel. L'interaction non en temps réel permet à l'utilisateur d'entrer des informations sur le fonctionnement d'application de façon asynchrone par rapport à son invocation. Fréquemment, ceci est fait par des systèmes de provisionnement. Par exemple, un utilisateur peut établir le numéro de transmission pour un transfert d'appel sur une application de non réponse en utilisant une page de la Toile. Une interaction en temps réel exige que l'utilisateur interagisse avec l'application au moment de son invocation.

4.3 Client local contre client distant

Un autre axe de taxonomie est si l'interface d'utilisateur est co résident avec l'appareil utilisateur (qu'on appelle une interface d'utilisateur de client local) ou si l'interface d'utilisateur fonctionne dans un hôte séparé du client (qu'on appelle une interface d'utilisateur de client distant). Dans une interface d'utilisateur de client distant, il existe une sorte de protocole entre l'appareil client et l'UI qui permet au client d'interagir avec l'interface d'utilisateur sur un réseau.

La façon la plus importante de séparer l'UI et l'appareil client est par l'interaction des supports. Dans l'interaction des supports, l'interface entre l'utilisateur et l'interface d'utilisateur est par le support : audio, vidéo, messagerie, et ainsi de suite. C'est le mode de fonctionnement classique pour VoiceXML [Voice-XML], où l'interface d'utilisateur (aussi appelé le navigateur vocal) fonctionne sur une plate-forme dans le réseau. Les utilisateurs communiquent avec le navigateur vocal par le réseau téléphonique (ou en utilisant une session SIP). Le navigateur vocal interagit avec l'application en utilisant HTTP pour porter les informations collectées de l'utilisateur.

Dans le cas d'une interface d'utilisateur de client local, l'interface d'utilisateur fonctionne colocalisée avec l'appareil d'utilisateur. L'interface entre eux est par le logiciel qui interprète les entrées de l'utilisateur et les passe à l'interface d'utilisateur. L'exemple classique de cela est la Toile. Sur la Toile, l'interface d'utilisateur est un navigateur, et l'interface est définie par le document HTML qu'elle rend. L'utilisateur interagit directement avec l'interface d'utilisateur fonctionnant dans le navigateur. Les résultats de cette interface d'utilisateur sont envoyés à l'application (qui fonctionne sur le serveur de la Toile) en utilisant HTTP.

Il est important de noter que si l'interface d'utilisateur est locale ou distante (dans le cas d'une interaction de supports) n'est pas une propriété de la modalité de l'interface, mais plutôt une propriété du système. Par exemple, il est possible à une interface d'utilisateur fondée sur la Toile d'être fournie avec une interface d'utilisateur de client distant. Dans ce scénario, les sessions de supports de partage de vidéo et d'application peuvent être utilisées entre l'utilisateur et l'interface d'utilisateur. L'interface d'utilisateur, toujours guidée par HTML, fonctionne maintenant "dans le réseau", à distance du client. De même, un document VoiceXML peut être interprété localement par un appareil client, sans flux de supports du tout. Bien sûr, le document VoiceXML peut être rendu en utilisant "text", plutôt que "media", sans impact sur l'interface entre l'interface d'utilisateur et l'application.

Il est aussi important de noter que les systèmes peuvent être hybrides. Dans une interface d'utilisateur hybride, certains de ses aspects (généralement ceux associés à une modalité particulière) fonctionnent en local, et d'autres à distance.

4.4 Capable de présentation contre sans présentation

Une interface d'utilisateur peut être capable de présenter l'information à l'utilisateur (un UI capable de présentation) ou peut être capable seulement de collecter les entrées d'utilisateur (un UI sans présentation). Ce sont des types d'interfaces d'utilisateur très différents. Un UI capable de présentation peut fournir des retours à l'utilisateur après chaque entrée, fournissant le contexte pour collecter la prochaine entrée. Par suite, les interfaces d'utilisateur capables de présentation exigent une mise à jour des informations fournies à l'utilisateur après chaque entrée. La Toile est un exemple classique de cela. Après chaque entrée (c'est-à-dire, un clic) le navigateur fournit l'entrée à l'application et va chercher la prochaine page à rendre. Dans une interface d'utilisateur sans présentation, ce n'est pas le cas. Comme aucun retour n'est fourni à l'utilisateur, ces interfaces d'utilisateur tendent à simplement collecter l'information comme elle est entrée, et la passer à l'application.

Une autre différence est qu'une interface d'utilisateur sans présentation ne peut pas prendre facilement en charge le concept d'une cible. Le choix d'une cible exige généralement un moyen pour informer l'utilisateur des applications disponibles, permettant à l'utilisateur de faire un choix, et ensuite de les informer de celle qu'il a choisi. Sans la première et la troisième étapes (qu'un UI sans présentation ne peut pas fournir) le choix d'une cible est très difficile. Sans choix d'une cible, l'entrée fournie aux applications par les interfaces d'utilisateur sans présentation est plus une opération de diffusion ou notification.

5. Scénarios d'interaction en téléphonie

Dans cette Section, on applique le modèle de la Section 4 aux téléphones.

Dans un téléphone traditionnel, l'interface d'utilisateur consiste en un clavier de 12 clés, un haut-parleur, et un microphone. Bien sûr, à partir d'ici, le terme "téléphone" est utilisé pour représenter tout appareil qui satisfait, au minimum, les caractéristiques décrites dans la phrase précédente. Les applications de téléphonie à commutation de circuit sont presque universellement des interfaces d'utilisateur de client distant. Dans le réseau téléphonique public commuté (RTPC) il y a généralement une interface de circuit entre l'utilisateur et l'interface d'utilisateur. L'entrée d'utilisateur à partir du clavier est convoyée en utilisant la bi tonalités multi fréquences (DTMF, *Dual-Tone Multi-Frequency*) et l'entrée du microphone comme de la voix codée en modulation par impulsions codées (MIC en français, PCM, *Pulse Code Modulated* en anglais).

Dans un système fondé sur IP, il y a plus de variabilité dans la façon dont le système peut être instancié. Les deux interfaces d'utilisateur de client distant et de client local à un téléphone peuvent être fournies.

Dans ce cadre, une passerelle RTPC peut être considérée comme un mandataire d'appareil d'utilisateur. C'est un mandataire pour l'utilisateur parce que il peut fournir, à une interface d'utilisateur sur un réseau IP, une entrée prise d'un utilisateur sur un téléphone à commutation de circuit. La passerelle peut être capable de faire fonctionner une interface d'utilisateur de client local, tout comme pourrait le faire un téléphone IP.

5.1 Client distant

L'instanciation la plus évidente est le modèle "classique" de téléphonie à commutation de circuit. Dans ce modèle, l'interface d'utilisateur fonctionne à distance du client. L'interface entre l'utilisateur et l'interface d'utilisateur est par des supports qui sont établis par SIP et portés sur le protocole de transport en temps réel (RTP, *Real Time Transport Protocol*) [RFC3550]. L'entrée du microphone peut être portée en utilisant tout algorithme de codage vocal convenable. L'entrée de clavier peut être convoyée d'une des deux façons suivantes : la première est de convertir l'entrée de clavier en DTMF, et ensuite de convoyer ce DTMF en utilisant un algorithme de codage convenable (comme le MIC loi µ). Une autre approche, généralement préférée, est de transmettre l'entrée du clavier en utilisant la [RFC4733], qui fournit un mécanisme de codage pour porter l'entrée de clavier dans RTP.

Dans ce modèle classique, l'interface d'utilisateur va fonctionner sur un serveur dans le réseau IP. Elle va effectuer la reconnaissance de la parole et la reconnaissance de DTMF pour déduire l'intention de l'utilisateur, les introduire par l'interface d'utilisateur, et fournir le résultat à une application.

5.2 Client local

Un autre modèle est que l'interface d'utilisateur entière réside sur le téléphone. L'interface d'utilisateur peut être un navigateur VoiceXML, effectuant la reconnaissance de parole sur l'entrée de microphone, et fournissant l'entrée de clavier directement dans le script. Comme discuté ci-dessus, le script VoiceXML pourrait être rendu en utilisant du texte au lieu de la voix, si le téléphone a un affichage textuel.

Pour des téléphones plus simples sans affichage, l'interface d'utilisateur peut être décrite par un document de demande du langage de balisage à pression de touche (KPML, *Keypad Markup Language*) [RFC4730]. Lorsque l'utilisateur entre les chiffres sur le clavier, ils sont passés à l'interface d'utilisateur, qui génère des événements d'interface d'utilisateur pouvant être transportés à l'application.

5.3 Bascule

Une approche intermédiaire est de faire des allers-retours entre une interface d'utilisateur de client local et de client distant. De nombreuses applications vocales sont du type qui écoute le flux de supports et attendent un déclencheur spécifique qui lance une interaction d'utilisateur plus complexe. La touche pause dans une application de carte d'appel prépayée est un exemple. Un autre exemple est une application d'enregistrement de conférence, où l'utilisateur peut presser une clé à un moment de l'appel pour commencer l'enregistrement. Quand la clé est pressée, l'utilisateur entend un chuchotement qui l'informe que l'enregistrement a commencé.

La façon idéale de prendre en charge de telles applications est d'installer un composant d'interface d'utilisateur de client local qui attend le déclencheur pour lancer l'interaction réelle. Une fois le déclencheur reçu, l'application connecte l'utilisateur à une interface d'utilisateur de client distant qui peut faire des annonces, collecter plus d'informations, etc.

L'avantage des allers-retours entre une interface d'utilisateur de client local et de client distant est le coût. L'interface d'utilisateur de client local va éliminer le besoin d'envoyer des flux de supports dans le réseau juste pour attendre que l'utilisateur presse la touche dièse sur le clavier.

Le langage de balisage de clavier (KPML, *Keypad Markup Language*) a été conçu pour prendre en charge exactement ce type de besoins [RFC4730]. Il modélise le clavier sur un téléphone et permet à une application d'être informée de toute séquence de clés pressée. Cependant, KPML n'a pas de composant de présentation. Comme les interfaces d'utilisateur exigent généralement une réponse à l'entrée de l'utilisateur, la présentation va devoir être faite en utilisant une interface d'utilisateur de client distant qui va être instanciée par suite du déclenchement.

Il est tentant d'utiliser un modèle hybride, où une application d'invite et collecte est mise en œuvre en utilisant une interface d'utilisateur de client distant qui exécute l'invite, et une interface d'utilisateur de client local, décrite par KPML, qui collecte les chiffres. Cependant, cela complique seulement l'application. D'abord, l'entrée de clavier va être envoyée à la fois au flux

de support et à l'interface d'utilisateur KPML. Cela exige que l'application trie quelles entrées d'utilisateur sont dupliquées, un processus qui est très compliqué. Ensuite, le principal avantage de KPML est d'éviter d'avoir un flux de supports en face d'une interface d'utilisateur. Cependant, il y a déjà un flux de supports pour l'invite, donc il n'y a pas de réelle économie

6. Vue d'ensemble du cadre

Dans ce cadre, on utilise le terme "application SIP" pour se référer à un large ensemble de fonctionnalités. Une application SIP est un programme fonctionnant sur un élément fondé sur SIP (comme un mandataire ou agent d'utilisateur) qui fournit une fonction de valeur ajoutée à un utilisateur ou administrateur de système. Les applications SIP peuvent s'exécuter au nom d'un appelant, d'un appelé, ou d'une multitude d'utilisateurs à la fois.

Chaque application a un certain nombre d'instances qui s'exécutent à tout moment. Une instance représente un seul chemin d'exécution pour une application. Elle est établie par suite d'un certain événement. Cet événement peut être un événement SIP, comme la réception d'une demande INVITE de SIP, ou elle peut être un événement non SIP, comme l'envoi d'un formulaire de la Toile ou même un temporisateur. Des instances d'application ont aussi une fin. Certaines instances ont une durée de vie qui est couplée avec une transaction ou dialogue SIP. Par exemple, une application mandataire pourrait commencer quand un INVITE arrive, et se terminer quand il est répondu à l'appel. D'autres applications ont une durée de vie qui s'étend sur plusieurs dialogues ou transactions. Par exemple, une instance d'application de conférence peut exister tant qu'il y a des dialogues qui lui sont connectés. Quand le dernier dialogue se termine, l'instance d'application se termine. D'autres applications ont une durée de vie qui est complètement découplée des événements SIP.

Il est fondamental pour le cadre décrit ici que plusieurs instances d'application puissent interagir avec un utilisateur durant une seule transaction ou dialogue SIP. Chaque instance peut être pour la même application, ou des applications différentes. Chaque application peut être complètement indépendante, en ce que chacune peut être possédée par un fournisseur différent, et peut ignorer l'existence de chaque autre. De même, il peut y avoir des instances d'application qui interagissent avec l'appelant, et des instances qui interagissent avec l'appelé, toutes deux dans la même transaction ou dialogue.

La première étape de l'interaction avec l'utilisateur est d'instancier un ou plusieurs composants d'interface d'utilisateur pour l'instance d'application. Un composant d'interface d'utilisateur est un seul élément de l'interface d'utilisateur qui est défini par un flux logique qui n'est pas couplé de façon synchrone avec un autre composant. En d'autres termes, chaque composant fonctionne indépendamment.

Un composant d'interface d'utilisateur peut être instancié dans un des agents d'utilisateur dans un dialogue (pour une interface d'utilisateur de client local) ou au sein d'un élément de réseau (pour une interface d'utilisateur de client distant). Si une interface d'utilisateur de client local est utilisée, l'application a besoin de déterminer si l'agent d'utilisateur est ou non capable de prendre en charge une interface d'utilisateur de client local, et dans quel format. Dans ce cadre, tous les composants d'interface d'utilisateur de client local sont décrits par un langage de balisage. Un langage de balisage décrit un flux logique de présentation des informations à l'utilisateur, une collection d'informations provenant de l'utilisateur, et une transmission de ces informations à une application. Les exemples de langages de balisage incluent HTML, le langage de balisage sans fil (WML, *Wireless Markup Language*), VoiceXML, et le langage de balisage à pression de touche (KPML, *Keypad Markup Language*) [RFC4730].

À la différence d'une instance d'application, qui a une durée de vie très souple, un composant d'interface d'utilisateur a une durée de vie fixe. Un composant d'interface d'utilisateur est toujours associé à un dialogue. Le composant d'interface d'utilisateur peut être créé à tout moment après la création du dialogue (ou du dialogue précoce). Cependant, le composant d'interface d'utilisateur se termine quand le dialogue se termine. Le composant d'interface d'utilisateur peut être terminé plus tôt par l'agent d'utilisateur, et éventuellement par l'application, mais sa durée de vie n'excède jamais celle de son dialogue associé.

Il y a deux façons de créer un composant d'interface de client local. Pour les composants d'interface qui sont capables de présentation, l'application envoie une demande REFER [RFC3515] à l'agent d'utilisateur. Le champ d'en-tête Refer-To contient un URI HTTP qui pointe sur le balisage de l'interface d'utilisateur, et le REFER contient un champ d'en-tête Target-Dialog [RFC4538] qui identifie le dialogue associé au composant d'interface d'utilisateur. Pour les composants d'interface d'utilisateur qui sont sans présentation (comme ceux définis par KPML) l'application envoie une demande SUBSCRIBE à l'agent d'utilisateur. Le corps de la demande SUBSCRIBE contient un filtre, qui, dans ce cas, est le balisage qui définit quand les informations sont à envoyer à l'application dans un NOTIFY. Le SUBSCRIBE ne contient pas de champ d'en-tête Target-Dialog, car des informations équivalentes sont portées dans le champ d'en-tête Event.

Si un composant d'interface d'utilisateur va être instancié dans le réseau, il n'y a pas besoin de déterminer les capacités de l'appareil sur lequel l'interface d'utilisateur est instanciée. On suppose que c'est sur un appareil dont l'application sait qu'un UI peut être créé. Cependant, l'application a besoin de connecter l'appareil de l'utilisateur à l'interface d'utilisateur. Cela va exiger la manipulation des flux de supports afin d'établir cette connexion.

L'interface entre le composant d'interface d'utilisateur et l'application dépend du type de l'interface d'utilisateur. Pour les interfaces d'utilisateur capables de présentation, comme celles décrites par HTML et VoiceXML, les opérations POST de forme HTTP sont utilisées. Pour les interfaces d'utilisateur sans présentation, un NOTIFY SIP est utilisé. Les différents besoins et capacités de ces deux interfaces d'utilisateur, comme décrit au paragraphe 4.4, sont ce qui dirige les différents choix pour les interactions. Comme les interfaces d'utilisateur capables de présentation exigent une mise à jour de la présentation chaque fois que des données d'utilisateur sont entrées, cela correspond bien à HTTP. Comme les interfaces d'utilisateur sans présentation transmettent simplement l'entrée d'utilisateur à l'application, un NOTIFY est plus approprié.

Bien sûr, pour les interfaces d'utilisateur sans présentation, il y a deux différentes modalités de fonctionnement. la première est appelée "à un coup". Dans le rôle à un coup, le balisage attend qu'un utilisateur entre des informations et, quand il le fait, rapporte cet événement à l'application. L'application fait alors quelque chose, et le balisage n'est plus utilisé. Dans l'autre modalité, appelée "surveillance", le balisage reste en résidence permanente, et rapporte les informations à une application jusqu'à la terminaison du dialogue associé.

7. Topologies de déploiement

Cette Section présente des topologies de réseau dans lesquelles ce cadre peut être instancié.

7.1 Application tierce

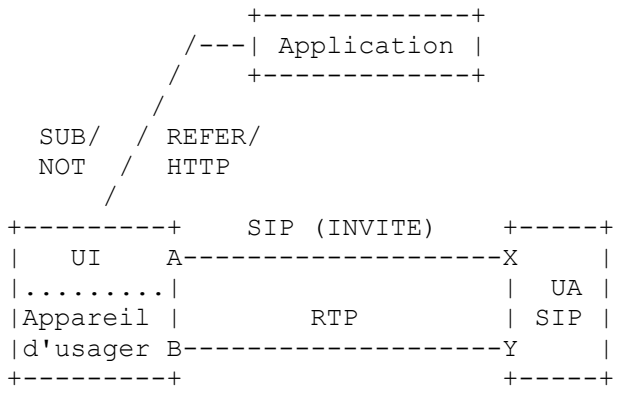


Figure 2 : Topologie tierce

Dans cette topologie, l'application qui est intéressée à interagir avec les utilisateurs existe en dehors du dialogue SIP entre les agents d'utilisateur. Dans ce cas, l'application apprend l'initiation et la terminaison du dialogue, avec les identifiants de dialogue, par des moyens hors bande. Une possibilité est que le paquetage d'événements de dialogue [RFC4235]. Les informations de dialogue ne sont révélées qu'aux parties de confiance, de sorte que l'application va devoir être de confiance pour un des utilisateurs afin d'obtenir ces informations.

À tout moment durant le dialogue, l'application peut instancier des composants d'interface d'utilisateur sur l'appareil d'utilisateur de l'appelant ou de l'appelé. Il peut le faire en utilisant SUBSCRIBE ou REFER, selon le type d'interface d'utilisateur (capable de présentation ou sans présentation).

7.2 Application co-résidente

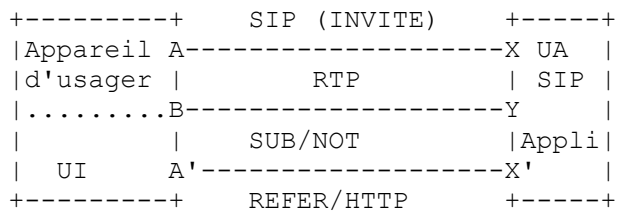


Figure 3 : Topologie co-résidente

Dans cette topologie, l'application est co-résidente avec un des agents d'utilisateur (celui qui est à droite de la figure). Cette application peut installer des composants d'interface d'utilisateur de client local sur l'autre agent d'utilisateur, qui agit comme appareil d'utilisateur. Ces composants peuvent être installés en utilisant soit SUBSCRIBE, pour des interfaces

d'utilisateur sans présentation, soit REFER, pour ceux capables de présentation. Cette situation survient normalement quand l'application souhaite installer des composants d'UI sur une interface d'utilisateur capable de présentation. Si la seule entrée d'utilisateur est via le clavier, le cadre n'est pas nécessaire par lui-même, parce que l'UA/application va recevoir l'entrée via la RFC 4733 dans le flux RTP.

Si l'application réside dans l'appelé, elle est appelée une "application de terminaison". Si elle réside chez l'appelant, elle est appélée une "application génératrice".

Cette sorte de topologie est courante dans les convertisseurs de protocole et les applications de passerelle.

7.3 Application tierce et mandataire d'appareil d'utilisateur

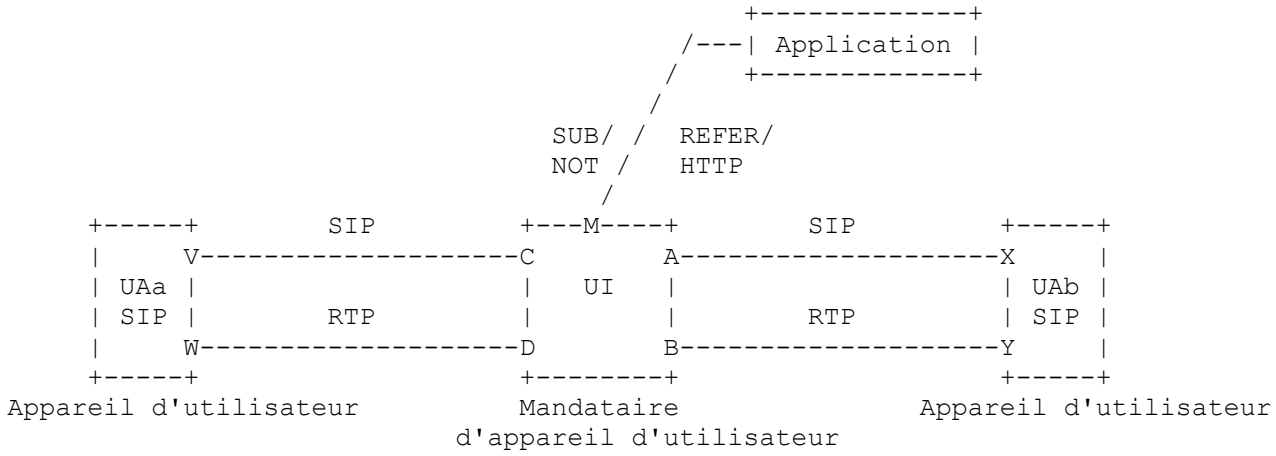
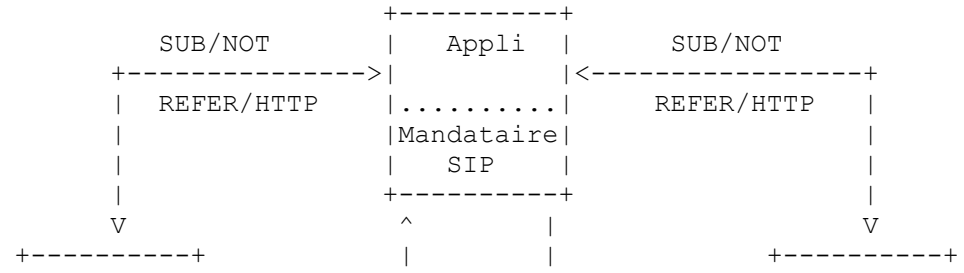


Figure 4 : Topologie de mandataire d'appareil d'utilisateur

Dans cette topologie, il y a une application tierce comme au paragraphe 7.1. Cependant, au lieu d'installer un composant d'interface d'utilisateur sur l'appareil d'utilisateur d'extrémité, le composant est installé dans un appareil intermédiaire, appelé un mandataire d'appareil d'utilisateur. Du point de vue de l'appareil utilisateur réel (à gauche) le mandataire d'appareil d'utilisateur est une interface d'utilisateur de client distant. À ce titre, les supports, normalement transportés en utilisant RTP (y compris la RFC 4733 pour porter l'entrée d'utilisateur) sont envoyés de l'appareil d'utilisateur à l'interface d'utilisateur de client distant sur le mandataire d'appareil d'utilisateur. Pour autant que l'application soit concernée, elle installé ce qu'elle pense être une interface d'utilisateur de client local sur l'appareil de l'utilisateur, mais il se trouve que c'est un mandataire d'appareil utilisateur qui ressemble à l'appareil d'utilisateur pour l'application.

Le mandataire d'appareil d'utilisateur va devoir terminer et re-générer la signalisation (SIP) et le trafic de supports vers les homologues réels de la conversation. Le mandataire d'appareil d'utilisateur est un relais de supports dans la terminologie de la [RFC3550]. Le mandataire d'appareil d'utilisateur va devoir surveiller le flux de supports associé à chaque dialogue, afin de convertir l'entrée d'utilisateur reçue dans le flux de supports en événements rapportés à l'interface d'utilisateur. Cela peut lancer un défi dans les systèmes multi-média, où le flux de supports où l'entrée d'utilisateur est envoyé peut ne pas apparaître clairement. Comme discuté dans la [RFC3264], si un agent d'utilisateur a une seule source de supports et prend en charge plusieurs flux, il est supposé envoyer cette source à tous les flux. Dans les cas où il y a plusieurs sources, la transposition est l'affaire de la politique locale. En l'absence d'un moyen d'identifier explicitement ou de demander quelles sources se transposent en quels flux, le mandataire d'appareil d'utilisateur va devoir faire du mieux qu'il peut. La présente spécification RECOMMANDE que le mandataire d'appareil d'utilisateur surveille le premier flux (défini en termes d'ordre des sessions de supports au sein d'une description de session). À ce titre, les agents d'utilisateur DEVRAIENT envoyer leur entrée d'utilisateur sur le premier flux, en l'absence d'une politique pour décider autrement.

7.4 Application mandataire



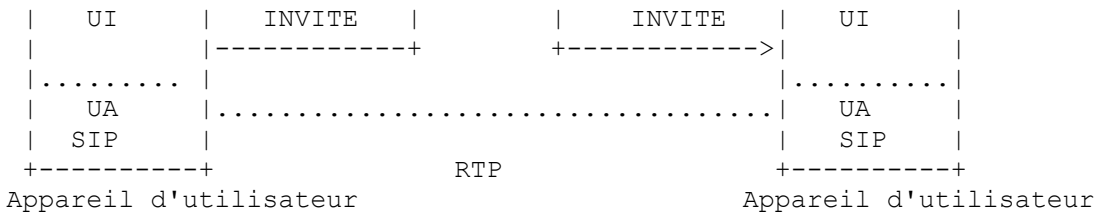


Figure 5 : Topologie d'application mandataire

Dans cette topologie, l'application est co-résidente avec un serveur mandataire de transaction à états pleins, qui enregistre l'acheminement sur le chemin d'appel entre deux appareils d'utilisateur. L'application utilise SUBSCRIBE ou REFER pour installer les composants d'interface d'utilisateur sur un appareil d'utilisateur ou les deux. Cette topologie est courante dans les applications d'acheminement, comme une application d'acheminement d'appel assistée par la Toile.

8. Comportement d'application

Le comportement d'une application au sein de ce cadre dépend de si elle cherche à utiliser une des interfaces d'utilisateur de client local ou de client distant.

8.1 Interfaces de client local

Un composant clé de ce cadre est la prise en charge des interfaces d'utilisateur de client local.

8.1.1 Découverte des capacités

Une interface d'utilisateur de client local peut seulement être instanciée sur un agent d'utilisateur si l'agent d'utilisateur prend en charge ce type de composant d'interface d'utilisateur. La prise en charge de composants d'interface d'utilisateur de client local est déclarée par l'UAC et l'UAS dans leurs champs d'en-tête Allow, Accept, Supported, et Allow-Event de demandes et réponses d'initiation de dialogue. Si le champ d'en-tête Allow indique la prise en charge de la méthode SIP SUBSCRIBE, et si le champ d'en-tête Allow-Event indique la prise en charge du paquetage KPML [RFC4730], et si le champ d'en-tête Supported indique la prise en charge de la spécification d'URI d'agent d'utilisateur mondialement acheminable (GRUU, *Globally Routable UA URI*) [RFC5627] (ce qui signifie que le champ d'en-tête Contact contient un GRUU) cela signifie que l'UA peut instancier des composants d'interface d'utilisateur sans présentation. Dans ce cas, l'application peut pousser des composants d'interface d'utilisateur sans présentation conformément aux règles du paragraphe 8.1.2. Les langages de balisage spécifiques qui peuvent être pris en charge sont indiqués dans le champ d'en-tête Accept.

Si le champ d'en-tête Allow indique la prise en charge de la méthode SIP REFER, et si le champ d'en-tête Supported indique la prise en charge du champ d'en-tête Target-Dialog [RFC4538], et si le champ d'en-tête Contact contient les capacités d'UA [RFC3840] qui indiquent la prise en charge du schéma d'URI HTTP, cela signifie que l'UA prend en charge les composants d'interface d'utilisateur capables de présentation. Dans ce cas, l'application peut pousser les composants d'interface d'utilisateur capables de présentation au client conformément aux règles du paragraphe 8.1.2. Les balisages spécifiques pris en charge sont indiqués dans le champ d'en-tête Accept.

Une application tierce qui n'est pas présente sur le chemin d'appel ne va pas avoir connaissance de ces champs d'en-tête dans les demandes d'initiation de dialogue qui passent. À ce titre, elle va devoir obtenir ces informations de capacités par d'autres moyens. Un moyen est par le paquetage d'événements d'enregistrement [RFC3680], qui peut contenir les informations de capacités de l'agent d'utilisateur fournies dans les demandes REGISTER [RFC3840].

8.1.2 Poussée d'un composant initial d'interface

Généralement, on prévoit que les composants d'interface vont devoir être créés à divers points différents dans une session SIP. En clair, ils vont devoir être poussés durant l'établissement de session, ou après l'établissement de la session. Un composant d'interface d'utilisateur est cependant toujours associé à un dialogue spécifique.

Une application NE DOIT PAS tenter de pousser un composant d'interface d'utilisateur à un agent d'utilisateur avant qu'il ait déterminé que l'agent d'utilisateur a les capacités nécessaires et qu'un dialogue a été créé. Dans le cas d'un UAC, cela signifie qu'une application NE DOIT PAS pousser un composant d'interface d'utilisateur pour un dialogue initié par INVITE jusqu'à ce que l'application ait vu une demande confirmant la réception d'une réponse de création de dialogue. Cela pourrait être un ACK pour une 200 OK, ou un PRACK pour une réponse provisoire [RFC3262]. Pour les dialogues initiés par SUBSCRIBE, l'application NE DOIT PAS pousser un composant d'interface d'utilisateur tant que l'application

n'a pas vu une 200 OK à la demande NOTIFY. Pour un composant d'interface d'utilisateur sur un UAS, l'application NE DOIT PAS pousser un composant d'interface d'utilisateur pour un dialogue initié par INVITE tant qu'il n'a pas vu une réponse de création de dialogue provenant de l'UAS. Pour un dialogue initié par SUBSCRIBE, il NE DOIT PAS pousser un composant d'interface d'utilisateur tant qu'il n'a pas vu une demande NOTIFY provenant du notificateur.

Pour créer un composant d'UI capable de présentation sur l'UA, l'application envoie une demande REFER à l'UA. Ce REFER DOIT être envoyé au GRUU [RFC5627] annoncé par cet UA dans le champ d'en-tête Contact de la demande ou réponse d'initiation de dialogue envoyée par cet UA. Noter que cette demande REFER crée un dialogue distinct entre l'application et l'UA. Le champ d'en-tête Refer-To de la demande REFER DOIT contenir un URI HTTP qui fait référence au document de balisage à aller chercher.

De plus, il est essentiel que la demande REFER soit corrélée avec le dialogue auquel le composant d'interface d'utilisateur va être associé. Ceci est nécessaire pour l'autorisation et pour terminer les composants d'interface d'utilisateur quand le dialogue se termine. Pour fournir ce contexte, la demande REFER DOIT contenir un champ d'en-tête Target-Dialog identifiant le dialogue auquel le composant d'interface d'utilisateur est associé. Comme exposé dans la [RFC4538], cette demande va aussi contenir un champ d'en-tête Require avec l'étiquette d'option "tdialog".

Pour créer un composant d'interface d'utilisateur sans présentation, l'application envoie une demande SUBSCRIBE à l'UA. Le SUBSCRIBE DOIT être envoyé au GRUU annoncé par l'UA. Cette demande SUBSCRIBE crée un dialogue séparé. La demande SUBSCRIBE DOIT utiliser le paquetage d'événements KPML [RFC4730]. Le corps de la demande SUBSCRIBE contient le document de balisage qui définit les conditions dans lesquelles l'application souhaite avoir notification des entrées d'utilisateur.

Dans les deux cas, la demande REFER ou SUBSCRIBE DEVRAIT inclure un nom d'affichage dans le champ d'en-tête From qui identifie le nom de l'application. Par exemple, une carte d'appel pré payée pourrait inclure un champ d'en-tête From qui ressemble à :

From: "Carte d'appel pré payée" <sip:prepaid@exemple.com>

Tous les mécanismes d'assertion d'identité SIP qui ont été définis, comme dans les [RFC3325] et [RFC4474], sont applicables aussi à ces demandes.

8.1.3 Mise à jour d'un composant d'interface

Une fois qu'un composant d'interface d'utilisateur a été créé sur un client, il peut être mis à jour. Le moyen pour le mettre à jour dépend du type de composant d'UI.

Les composants d'UI capables de présentation sont mis à jour en utilisant des techniques déjà en place pour ces balisages. En particulier, l'entrée d'utilisateur va causer une opération HTTP POST pour pousser l'entrée d'utilisateur à l'application. Le résultat de l'opération POST est un nouveau balisage que l'UI est supposé utiliser. Cela permet à l'UI d'être mis à jour en réponse à l'action de l'utilisateur. Certains balisages, comme HTML, donnent la capacité de forcer un rafraîchissement après un certain temps, afin que l'UI puisse être mis à jour sans entrée de l'utilisateur. Ces mécanismes peuvent aussi être utilisés ici. Cependant, la poussée asynchrone d'un composant d'UI mis à jour de l'application à l'agent d'utilisateur n'est pas prise en charge. Une nouvelle demande REFER au même GRUU va créer un nouveau composant d'UI plutôt qu'une mise à jour de composants déjà en place.

Pour l'UI sans présentation, c'est différent. L'application PEUT mettre à jour le filtre à tout moment en générant un rafraîchissement de SUBSCRIBE avec le nouveau filtre. L'UA va immédiatement commencer à utiliser ce nouveau filtre.

8.1.4 Terminaison d'un composant d'interface

Les composants d'interface d'utilisateur ont une durée de vie bien définie. Ils sont créés quand le composant est poussé au client. Les composants d'interface d'utilisateur sont toujours associés au dialogue SIP sur lequel ils ont été poussé. À ce titre, leur durée de vie est liée à la durée de vie du dialogue. Quand le dialogue se termine, le composant d'interface en fait autant.

Cependant, il y a des cas où l'application aimerait terminer le composant d'interface d'utilisateur avant son point de terminaison naturel. Pour les interfaces d'utilisateur capables de présentation, cela n'est pas possible. Pour les interfaces d'utilisateur sans présentation, l'application PEUT terminer le composant en envoyant un SUBSCRIBE avec Expires égal à zéro. Cela termine l'abonnement, et supprime le composant d'UI.

Un client peut supprimer un composant d'UI à tout moment. Pour un UI capable de présentation, c'est analogue à ce que l'utilisateur ferme la fenêtre du formulaire de la Toile. Il n'y a pas de mécanisme pour rapporter cette sorte d'événement à

l'application. L'application DOIT être prête à arriver en fin de temporisation et ne jamais recevoir d'entrée d'un utilisateur. La durée de cette temporisation dépend de l'application. Pour les interfaces d'utilisateur sans présentation, l'UA peut explicitement terminer l'abonnement. Il va en résulter la génération d'un NOTIFY avec un champ d'en-tête Subscription-State égal à "terminated".

8.2 Interfaces de client distant

En remplacement de, ou conjointement avec des interfaces d'utilisateur de client local, une application peut utiliser des interfaces d'utilisateur de client distant. Ces interfaces d'utilisateur peuvent s'exécuter en co-résidence avec l'application elle-même (dans ce cas aucune interface normalisée entre l'UI et l'application n'a besoin d'être utilisée) ou elles peuvent fonctionner séparément. Ce cadre suppose que l'interface d'utilisateur fonctionne sur un hôte qui a une relation de confiance suffisante avec l'application. À ce titre, les moyens pour instancier l'interface d'utilisateur ne sont pas considérés ici.

Le principal problème est de connecter l'appareil d'utilisateur à l'interface d'utilisateur distant. Le faire exige la manipulation des flux de supports entre le client et l'interface d'utilisateur. Une telle manipulation peut seulement être faite par les agents d'utilisateur. Il y a deux types d'applications d'agent d'utilisateur au sein de ce cadre : applications d'origine/terminaison, et applications intermédiaires.

8.2.1 Applications d'origine et de terminaison

Les applications d'origine et de terminaison sont des applications qui sont elles mêmes le générateur ou le receveur final d'une invitation SIP. Elles sont de "pures" applications d'agent d'utilisateur, pas des agents d'utilisateur de boucle locale. L'exemple classique de telles applications est une application de réponse vocale interactive (IVR, *interactive voice response*) qui est normalement une application de terminaison. C'est une application de terminaison parce que l'utilisateur l'invoque explicitement ; c'est-à-dire, il est la partie appelée réelle. Un exemple d'application génératrice est une application d'appel de réveil, qui appelle un utilisateur à une heure spécifiée afin de l'éveiller.

Parce que les applications d'origine et de terminaison sont un point naturel de terminaison du dialogue, la manipulation de la session de supports par l'application est triviale. Les techniques traditionnelles de SIP pour ajouter et supprimer les flux de supports, modifier les codecs, et changer l'adresse du receveur du flux de supports peuvent être appliquées.

8.2.2 Applications intermédiaires

Les applications intermédiaires sont, en même temps, plus courantes que les applications d'origine/terminaison, et plus complexes. Les applications intermédiaires sont des applications qui ne sont ni l'appelant réel ni l'appelé. Elles représentent plutôt un "tiers" qui souhaite interagir avec l'utilisateur. L'exemple classique est l'application très répandue de carte d'appel prépayée.

Pour que l'application intermédiaire ajoute une interface d'utilisateur de client distant, elle doit manipuler le flux de supports de l'agent d'utilisateur pour se terminer sur cette interface d'utilisateur. Cela introduit aussi un problème fondamental d'interaction de caractéristiques. Comme l'application intermédiaire n'est pas un participant réel à l'appel, l'utilisateur va devoir interagir à la fois avec l'application intermédiaire et avec son homologue dans le dialogue. Faire les deux en même temps est compliqué et est discuté plus en détails à la Section 10.

9. Comportement d'agent d'utilisateur

9.1 Annonce des capacités

Afin de participer aux applications qui utilisent des interfaces de stimulation, un agent d'utilisateur a besoin d'annoncer ses capacités d'interaction.

Si un agent d'utilisateur prend en charge les interfaces d'utilisateur capables de présentation, il DOIT prendre en charge la méthode REFER. Il DOIT inclure, dans toutes les demandes et réponses d'initiation de dialogue, un champ d'en-tête Allow qui inclut la méthode REFER. L'agent d'utilisateur DOIT prendre en charge la spécification du dialogue cible [RFC4538], et DOIT inclure l'étiquette d'option "tdialog" dans le champ d'en-tête Supported des demandes et réponses de formation de dialogue. De plus, l'UA DOIT prendre en charge la spécification de capacités de l'agent d'utilisateur SIP [RFC3840]. L'UA DOIT être capable de traiter un REFER d'un URI HTTP. Il DOIT inclure, dans le champ d'en-tête Contact de ses demandes et réponses d'initiation de dialogue, un paramètre de champ d'en-tête Contact "schemes" qui inclut le schéma d'URI HTTP. L'UA DOIT inclure, dans toutes les demandes et réponses d'initiation de dialogue, un champ d'en-tête Accept faisant la liste de tous les balisages supportés par l'UA. Il est RECOMMANDÉ que tous les agents d'utilisateur qui prennent en charge les interfaces d'utilisateur capables de présentation prennent en charge HTML.

Si un agent d'utilisateur prend en charge les interfaces d'utilisateur sans présentation, il DOIT prendre en charge la méthode

SUBSCRIBE [RFC3265]. Il DOIT prendre en charge le paquetage d'événements KPML [RFC4730]. Il DOIT inclure, dans toutes les demandes et réponses d'initiation de dialogue, un champ d'en-tête Allow qui inclut la méthode SUBSCRIBE. Il DOIT inclure, dans toutes les demandes et réponses d'initiation de dialogue, un champ d'en-tête Allow-Events qui mentionne le paquetage d'événements KPML. L'UA DOIT inclure, dans toutes les demandes et réponses d'initiation de dialogue, un champ d'en-tête Accept qui fait la liste des filtres d'événement qu'il prend en charge . Au minimum, un UA DOIT prendre en charge le type MIME "application/kpml-request+xml".

Pour les interfaces d'utilisateur sans présentation ou capables de présentation, l'agent d'utilisateur DOIT prendre en charge la spécification de GRUU [RFC5627]. Le champ d'en-tête Contact dans toutes les demandes et réponses d'initiation de dialogue DOIT contenir un GRUU. L'UA DOIT inclure un champ d'en-tête Supported qui contient l'étiquette d'option "gruu" et l'étiquette d'option "tdialog".

Parce que ces en-têtes sont examinés par les mandataires qui peuvent exécuter les applications, un UA qui souhaite prendre en charge les interfaces d'utilisateur de client local ne devrait pas les chiffrer.

9.2 Réception des composants d'interface d'utilisateur

Une fois que l'UA a créé un dialogue (dans l'état précoce ou confirmé) il DOIT être prêt à recevoir une demande SUBSCRIBE ou REFER contre son GRUU. Si l'UA reçoit une telle demande avant l'établissement d'un dialogue, l'UA DOIT rejeter la demande.

Un agent d'utilisateur DEVRAIT tenter d'authentifier l'envoyeur de la demande. L'envoyeur va généralement être une application ; donc, il est peu probable que l'agent d'utilisateur ait un secret partagé avec lui, rendant l'authentification par résumé inutile. Cependant, des identités authentifiées peuvent être obtenues par d'autres moyens, comme le mécanisme Identity RFC4474].

Un agent d'utilisateur PEUT avoir des politiques d'autorisation pré-définies qui permettent aux applications qui se sont authentifiées avec une identité particulière de pousser des composants d'interface d'utilisateur. Si un tel ensemble de politiques est présent, il est d'abord vérifié. Si l'application est autorisée, le traitement se fait.

Si l'application s'est authentifiée mais n'est pas explicitement autorisée ou est bloquée, la présente spécification RECOMMANDE que l'application soit automatiquement autorisée si elle peut prouver qu'elle a été sur le chemin d'appel, ou est de confiance pour un des éléments sur le chemin d'appel. Une application prouve cela à l'agent d'utilisateur en démontrant qu'elle connaît les identifiants de dialogue. Cela se fait en les incluant dans le champ d'en-tête Target-Dialog pour les demandes REFER, ou dans les paramètres de champ d'en-tête Event de la demande KPML SUBSCRIBE.

Parce que les identifiants de dialogue servent d'outil pour l'autorisation, un agent d'utilisateur conforme au présent cadre DEVRAIT utiliser des identifiants de dialogue aléatoires cryptographiquement, avec au moins 128 bits d'aléa. Il est recommandé que cet aléa soit partagé entre les étiquettes de champ d'en-tête Call-ID et From dans le cas d'un UAC.

De plus, pour assurer que seules les applications qui résident dans le chemin ou sont de confiance pour des éléments dans le chemin peuvent instancier un composant d'interface d'utilisateur, un agent d'utilisateur conforme à la présente spécification DEVRAIT utiliser le schéma d'URI du protocole d'initiation de session sûre (SIPS, *Session Initiation Protocol Secure*) pour tous les dialogues qu'il initie. Cela va garantir des liaisons sûres entre tous les éléments dans le chemin de signalisation.

Si le dialogue n'a pas été établi avec un URI SIPS, ou si l'agent d'utilisateur n'a pas choisi des identifiants de dialogue cryptographiquement aléatoires, alors l'application NE DOIT PAS être autorisée automatiquement, même si elle a présenté des identifiants de dialogue valides. Un agent d'utilisateur PEUT appliquer toute autre politique en plus de (mais pas à la place de) celles spécifiées ici afin d'autoriser la création du composant d'interface d'utilisateur. Un de ces mécanismes serait d'inviter l'utilisateur, en l'informant de l'identité de l'application et du dialogue auquel elle est associée. Si une politique d'autorisation exige une interaction d'utilisateur, l'agent d'utilisateur DEVRAIT répondre à la demande SUBSCRIBE ou REFER avec un 202. Dans le cas de SUBSCRIBE, si l'autorisation n'est pas accordée, l'agent d'utilisateur DEVRAIT générer un NOTIFY pour terminer l'abonnement. Dans le cas d'un REFER, l'agent d'utilisateur NE DOIT PAS agir sur l'URI dans le champ d'en-tête Refer-To tant que l'autorisation de l'utilisateur n'est pas obtenue.

Si une application ne présente pas un identifiant de dialogue valide dans sa demande REFER ou SUBSCRIBE, l'agent d'utilisateur DOIT rejeter la demande avec une réponse 403.

Si une demande REFER à un URI HTTP est autorisée, l'UA exécute l'URI et va chercher le contenu à rendre à l'utilisateur. Cela instancie un composant d'interface d'utilisateur capable de présentation. Si un SUBSCRIBE a été autorisé, un composant d'interface d'utilisateur sans présentation est instancié.

9.3 Transposition d'entrée d'utilisateur en composants d'interface d'utilisateur

Une fois que les composants d'interface d'utilisateur sont instanciés, l'agent d'utilisateur doit diriger l'entrée d'utilisateur sur le composant approprié. Dans le cas d'interfaces d'utilisateur capables de présentation, ce processus est appelé le choix de cible. Il est fait par des moyens spécifiques de l'interface d'utilisateur sur l'appareil. Dans le cas d'un ordinateur personnel, par exemple, le gestionnaire de fenêtre va permettre à l'utilisateur de choisir le composant d'interface d'utilisateur approprié où diriger l'entrée.

Pour les interfaces d'utilisateur sans présentation, la situation est plus compliquée. Dans certains cas, l'appareil peut prendre en charge un mécanisme permettant à l'utilisateur de choisir une "ligne", et donc le dialogue associé. Toute entrée d'utilisateur sur le clavier quand cette ligne est sélectionnée est fournie aux composants d'interface d'utilisateur associés à ce dialogue.

Autrement, pour les interfaces d'utilisateur de client local, l'entrée d'utilisateur est supposée être associée à tous les composants d'interface d'utilisateur. Pour les interfaces d'utilisateur de client distant, l'appareil d'utilisateur convertit l'entrée d'utilisateur en supports, portés normalement en utilisant la RFC 4733, et il envoie cela à l'interface d'utilisateur de client distant. Cette interface d'utilisateur a alors besoin de transposer l'entrée d'utilisateur provenant potentiellement de nombreux flux de supports en événements d'interface d'utilisateur. Le processus pour faire cela est décrit au paragraphe 7.3.

9.4 Réception des mises à jour des composants d'interface d'utilisateur

Pour les interfaces d'utilisateur capables de présentation, les mises à jour de l'interface d'utilisateur sont faites de façon spécifique de ce composant d'interface d'utilisateur. Dans le cas de HTML, par exemple, le document peut dire au client d'aller chercher périodiquement un nouveau document. Cependant, le présent cadre ne donne aucun mécanisme supplémentaire pour pousser de façon asynchrone un nouveau composant d'interface d'utilisateur au client.

Pour les interfaces d'utilisateur sans présentation, une application peut pousser une mise à jour d'un composant en envoyant un rafraîchissement de SUBSCRIBE avec un nouveau filtre. L'agent d'utilisateur va les traiter conformément aux règles du paquetage d'événements.

9.5 Terminaison d'un composant d'interface d'utilisateur

La terminaison d'un composant d'interface d'utilisateur capable de présentation est une procédure triviale. L'agent d'utilisateur supprime simplement la fenêtre (ou son équivalent). Le fait que le composant soit supprimé n'est pas communiqué à l'application. À ce titre, c'est une affaire purement locale.

Dans le cas d'une interface d'utilisateur sans présentation, l'utilisateur pourrait souhaiter cesser d'interagir avec l'application. Cependant, la plupart des interfaces d'utilisateur sans présentation n'auront pas de moyen pour que l'utilisateur signale cela à travers l'appareil. Si un tel mécanisme existe, l'UA DEVRAIT générer une demande NOTIFY avec un champ d'en-tête Subscription-State égal à "terminated" et une raison de "rejected". Cela dit à l'application que le composant a été supprimé et qu'elle ne devrait pas tenter de se réabonner.

10. Interaction de caractéristique inter-applications

Le problème de l'interaction de caractéristiques inter-applications est inhérent à la signalisation par stimulus. Chaque fois qu'il y a plusieurs applications, il y a plusieurs interfaces d'utilisateur. Le système doit déterminer à quelle interface d'utilisateur est destinée une entrée particulière. Cette question est l'essence du problème de l'interaction de caractéristique inter-applications.

L'interaction de caractéristique inter-applications n'est pas un problème facile à résoudre. Pour l'instant, on examinera séparément les questions pour les composants d'interface d'utilisateur de client local et de client distant.

10.1 UI de client local

Quand l'interface d'utilisateur réside elle-même localement sur l'appareil client, le problème de l'interaction de caractéristiques est en fait beaucoup plus simple. L'appareil d'extrémité connaît explicitement chaque application, et donc peut présenter chacune séparément à l'utilisateur. Quand l'utilisateur fournit l'entrée, l'appareil client peut déterminer à quelle interface d'utilisateur l'entrée est destinée. L'interface d'utilisateur à laquelle l'entrée est destinée est appelée "application visée", et les moyens par lesquels l'application visée est choisie est appelée "détermination de cible".

D'une façon générale, la détermination de cible est une opération purement locale. Dans l'univers de l'ordinateur personnel,

la détermination de cible est fournie par les gestionnaires de fenêtre. Aucune application ne connaît les cibles ; elle reçoit simplement l'entrée d'utilisateur qui lui a été ciblée quand elle est en vue. Ce concept de base s'applique aussi aux applications fondées sur SIP.

La détermination de cible va fréquemment être triviale, selon le type d'interface d'utilisateur. Considérons un utilisateur qui fait un appel à partir d'un PC. L'appel passe à travers une application de carte d'appel prépayée et une application d'enregistrement d'appel. Toutes deux souhaitent interagir avec l'utilisateur. Toutes deux poussent une interface d'utilisateur fondée sur HTML à l'utilisateur. Sur le PC, chaque interface d'utilisateur va apparaître comme une fenêtre séparée. L'utilisateur interagit avec l'application d'enregistrement d'appel en choisissant sa fenêtre, et avec l'application de carte d'appel prépayée en choisissant sa fenêtre. La détermination de cible est littéralement fournie par le gestionnaire de fenêtre du PC. L'application à laquelle est ciblée l'entrée d'utilisateur est claire.

Dans un autre exemple, considérons les deux mêmes applications, mais sur un "téléphone intelligent" qui a un ensemble de boutons, et près de chaque bouton, il y a un affichage à cristaux liquides (LCD, *Liquid Cristal Display*) qui peut fournir une option à l'utilisateur. Cette interface d'utilisateur peut être représentée en utilisant le langage de balisage sans fil (WLM, *Wireless Markup Language*) par exemple.

Le téléphone va allouer un certain nombre de boutons à chaque application. La carte d'appel prépauée va avoir un bouton pour sa commande "décrocher", et l'application d'enregistrement va en avoir un pour sa commande "marche/arrêt". L'utilisateur peut facilement déterminer avec quelle application interagir en pressant le bouton approprié. Presser un bouton détermine la cible et fournit en même temps l'entrée d'utilisateur.

Malheureusement, tous les appareils n'ont pas ces affichages évolués. Une passerelle RTPC, ou un téléphone IP de base, peuvent avoir seulement un clavier de douze touches. Les interfaces d'utilisateur pour ces appareils sont fournies par le langage de balisage de clavier (KPML, *Keypad Markup Language*). En considérant une fois encore le cas d'interaction de caractéristiques ci-dessus, l'application de carte d'appel prépayée et l'application d'enregistrement d'appel vont toutes deux passer un document KPML à l'appareil. Quand l'utilisateur presse un bouton sur le clavier, à quel document l'entrée s'applique t-elle ? L'appareil ne permet pas à l'utilisateur de choisir. Un appareil où l'utilisateur ne peut pas fournir de cible est appelé un "appareil sans cible". C'est un problème assez difficile à résoudre. Le présent cadre ne fait aucune recommandation normative explicite, mais il conclut que la meilleure option est d'envoyer l'entrée aux deux interfaces d'utilisateur sauf si le balisage dans une des interfaces a indiqué qu'elle devrait être supprimée des autres. Ceci est un choix raisonnable par analogie -- c'est exactement ce que le réseau téléphonique à commutation de circuit existant va faire. C'est un non but explicite de fournir un meilleur mécanisme pour la résolution d'interaction de caractéristiques que le RTPC sur les appareils qui ont la même interface d'utilisateur que sur le RTPC. Les appareils avec de meilleurs affichages, comme les ordinateurs personnels ou les téléphones à écran, peuvent bénéficier des capacités de ce cadre, permettant à l'utilisateur de déterminer avec quelle application ils interagissent.

Bien sûr, quand un utilisateur fournit une entrée sur un appareil sans cible, l'entrée doit être passée à toutes les interfaces d'utilisateur de client local ET à toutes les interfaces d'utilisateur de client distant, sauf si le balisage dit à l'UI de supprimer les supports. Dans le cas de KPML, les événements clés sont passés aux interfaces d'utilisateur distantes en les codant comme décrit dans la [RFC4733]. Bien sûr, comme un client ne peut pas déterminer si un flux de supports se termine ou non dans une interface d'utilisateur distant, ces événements clés sont passés dans tous les flux de supports audio sauf si le document de demande KPML est utilisé pour les supprimer.

10.2 UI de client distant

Quand les interfaces d'utilisateur fonctionnent à distance, la détermination de la cible peut être bien plus difficile. De nombreuses architectures peuvent être déployées pour traiter l'interaction. Aucune n'est idéale. Cependant, toutes sortent du domaine d'application de la présente spécification.

11. Interaction de caractéristique intra application

Une application peut instancier plusieurs composants d'interface d'utilisateur. Par exemple, une seule application peut instancier deux composants HTML séparés et un composant WML. De plus, une application peut instancier des interfaces d'utilisateur à la fois de client local et de client distant.

Les problèmes d'interaction de caractéristiques entre ces composants au sein de la même application sont moins sévères. Si une application a plusieurs composants de client d'interface d'utilisateur, leur interaction est résolue de façon identique à celle du cas d'inter applications -- par la détermination de la cible. Cependant, les problèmes dans les appareils d'utilisateur sans cible (comme un clavier de téléphone) ne vont généralement pas exister, car l'application peut générer des interfaces d'utilisateur qui ne se chevauchent pas dans leur usage d'une entrée.

Le problème réel est que l'expérience optimale d'utilisateur exige fréquemment une sorte de couplage entre les différents composants d'interface d'utilisateur. C'est le problème classique des interfaces d'utilisateur multi-modales, comme celles décrites par les étiquettes de langage d'application de parole (SALT, *Speech Application Language Tag*). Par exemple, considérons une interface d'utilisateur où un utilisateur peut soit presser un bouton étiqueté pour faire un choix, soit écouter une invite, et dire le choix désiré. Idéalement, quand l'utilisateur presse le bouton, l'invite devrait cesser immédiatement, car tous deux ont été ciblés à collecter la même information en parallèle. De telles interactions sont mieux traitées par les balisages qui prennent en charge de façon native de telles interactions, comme SALT, et donc n'exigent aucune prise en charge explicite pour ce cadre.

12. Exemple de flux d'appels

Cette section montre le fonctionnement d'une application d'enregistrement d'appel. Cette application permet à un utilisateur d'enregistrer le support dans l'appel en cliquant sur un bouton dans un formulaire de la Toile. L'application utilise un composant d'interface d'utilisateur capable de présentation qui est poussé par l'appelant. Les conventions de la [RFC4475] sont utilisées pour décrire la représentation des lignes longues de message.

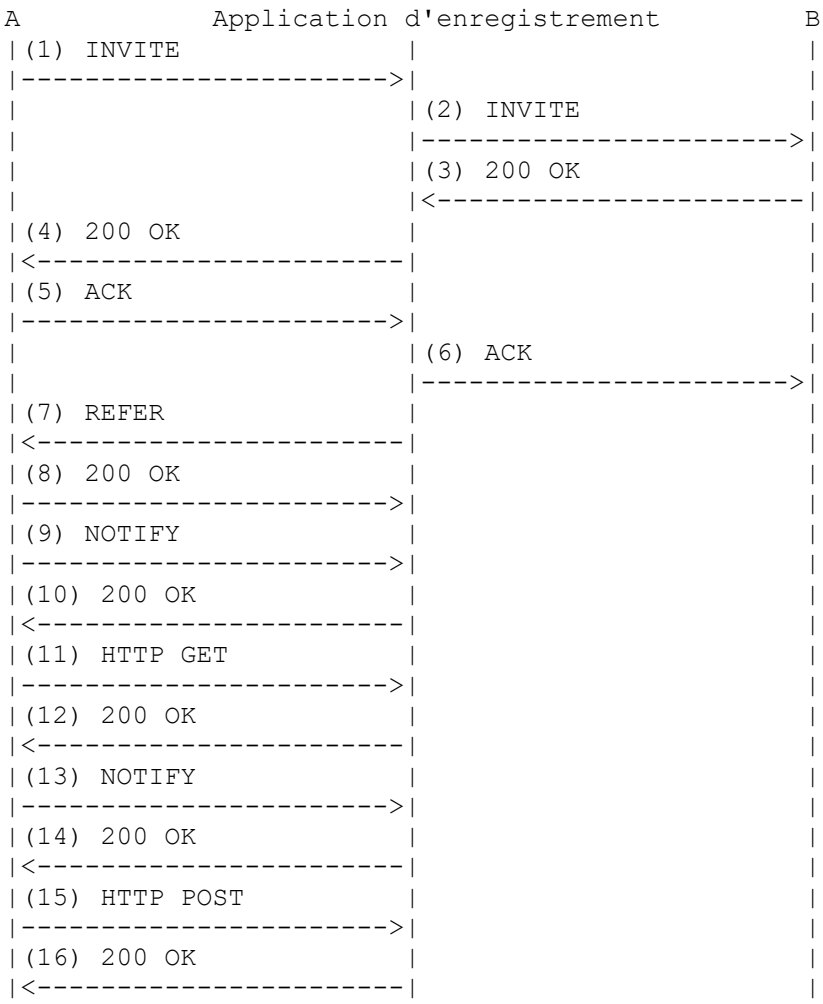


Figure 6

D'abord l'appelant, A, envoie un INVITE pour établir un appel (message 1). Comme l'appelant prend en charge le cadre et peut traiter les composants d'interface d'utilisateur capables de présentation, il inclut le champ d'en-tête Supported ce qui indique que l'extension GRUU et le champ d'en-tête Target-Dialog sont compris, le champ d'en-tête Allow indiquant que REFER est compris, et le champ d'en-tête Contact qui inclut le paramètre de champ d'en-tête "schemes".

INVITE sip:B@exemple.com SIP/2.0
Via: SIP/2.0/TLS host.exemple.com;branch=z9hG4bK9zz8
From: Caller <sip:A@exemple.com>;tag=kkaz-
To: Callee <sip:B@exemple.org>
Call-ID: fa77as7dad8-sd98ajzz@host.exemple.com
CSeq: 1 INVITE

```
Max-Forwards: 70
Supported: gruu, tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, REFER
Accept: application/sdp, text/html
<allOneLine>
Contact: <sip:A@exemple.com;gr=urn:uuid:f81d4fae
-7dec-11d0-a765-00a0c91e6bf6>;schemes="http,sip"
</allOneLine>
Content-Length: ...
Content-Type: application/sdp

--SDP non montré--
```

Le mandataire agit comme serveur d'enregistrement, et transmet l'INVITE à l'appelé (message 2). Il supprime le Record-Route qu'il insérerait normalement à cause de la présence du GRUU dans l'INVITE :

```
INVITE sip:B@pc.exemple.com SIP/2.0
Via: SIP/2.0/TLS app.exemple.com;branch=z9hG4bK97sh
Via: SIP/2.0/TLS host.exemple.com;branch=z9hG4bK9zz8
From: Caller <sip:A@exemple.com>;tag=kkaz-
To: Callee <sip:B@exemple.org>
Call-ID: fa77as7dad8-sd98ajzz@host.exemple.com
CSeq: 1 INVITE
Max-Forwards: 70
Supported: gruu, tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, REFER
Accept: application/sdp, text/html
<allOneLine>
Contact: <sip:A@exemple.com;gr=urn:uuid:f81d4fae
-7dec-11d0-a765-00a0c91e6bf6>;schemes="http,sip"
</allOneLine>
Content-Length: ...
Content-Type: application/sdp

--SDP non montré--
```

B accepte l'appel avec un 200 OK (message 3). Il ne prend pas en charge le cadre, de sorte que les divers champs d'en-tête ne sont pas présents.

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS app.exemple.com;branch=z9hG4bK97sh
Via: SIP/2.0/TLS host.exemple.com;branch=z9hG4bK9zz8
From: Caller <sip:A@exemple.com>;tag=kkaz-
To: Callee <sip:B@exemple.com>;tag=7777
Call-ID: fa77as7dad8-sd98ajzz@host.exemple.com
CSeq: 1 INVITE
Contact: <sip:B@pc.exemple.com>
Content-Length: ...
Content-Type: application/sdp

--SDP non montré--
```

Ce 200 OK est repassé à l'appelant (message 4) :

```
SIP/2.0 200 OK
Record-Route: <sip:app.exemple.com;lr>
Via: SIP/2.0/TLS host.exemple.com;branch=z9hG4bK9zz8
From: Caller <sip:A@exemple.com>;tag=kkaz-
To: Callee <sip:B@exemple.com>;tag=7777
Call-ID: fa77as7dad8-sd98ajzz@host.exemple.com
CSeq: 1 INVITE
Contact: <sip:B@pc.exemple.com>
Content-Length: ...
Content-Type: application/sdp
```

--SDP non montré--

L'appelant génère un ACK (message 5).

```
ACK sip:B@pc.exemple.com
Route: <sip:app.exemple.com;lr>
Via: SIP/2.0/TLS host.exemple.com;branch=z9hG4bK9zz9
From: Caller <sip:A@exemple.com>;tag=kkaz-
To: Callee <sip:B@exemple.com>;tag=7777
Call-ID: fa77as7dad8-sd98ajzz@host.exemple.com
CSeq: 1 ACK
```

Le ACK est transmis à l'appelé (message 6).

```
ACK sip:B@pc.exemple.com
Via: SIP/2.0/TLS app.exemple.com;branch=z9hG4bKh7s
Via: SIP/2.0/TLS host.exemple.com;branch=z9hG4bK9zz9
From: Caller <sip:A@exemple.com>;tag=kkaz-
To: Callee <sip:B@exemple.com>;tag=7777
Call-ID: fa77as7dad8-sd98ajzz@host.exemple.com
CSeq: 1 ACK
```

Maintenant, l'application décide de pousser un composant d'interface d'utilisateur à l'utilisateur A. Donc, il lui envoie une demande REFER (message 7) :

```
<allOneLine>
REFER sip:A@exemple.com;gr=urn:uuid:f81d4fae
-7dec-11d0-a765-00a0c91e6bf6 SIP/2.0
</allOneLine>
Refer-To: https://app.exemple.com/script.pl
Target-Dialog: fa77as7dad8-sd98ajzz@host.exemple.com
  ;remote-tag=7777;local-tag=kkaz-
Require: tdialog
Via: SIP/2.0/TLS app.exemple.com;branch=z9hG4bK9zh6
Max-Forwards: 70
From: Recorder Application <sip:app.exemple.com>;tag=jhgf
<allOneLine>
To: Caller <sip:A@exemple.com;gr=urn:uuid:f81d4fae
-7dec-11d0-a765-00a0c91e6bf6>
</allOneLine>
Require: tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, REFER
Call-ID: 66676776767@app.exemple.com
CSeq: 1 REFER
Event: refer
Contact: <sip:app.exemple.com>
```

Comme l'application enregistreuse est la même que le mandataire d'autorité pour le domaine, elle résout l'URI de demande en le contact enregistré de A, et l'y envoie ensuite. Il est répondu au REFER par un 200 OK (message 8).

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS app.exemple.com;branch=z9hG4bK9zh6
From: Recorder Application <sip:app.exemple.com>;tag=jhgf
To: Caller <sip:A@exemple.com>;tag=pqoew
Call-ID: 66676776767@app.exemple.com
Supported: gruu, tdialog
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, REFER
<allOneLine>
Contact: <sip:A@exemple.com;gr=urn:uuid:f81d4fae
-7dec-11d0-a765-00a0c91e6bf6>;schemes="http,sip"
</allOneLine>
CSeq: 1 REFER
```

L'utilisateur A envoie un NOTIFY (message 9) :

```
NOTIFY sip:app.exemple.com SIP/2.0
Via: SIP/2.0/TLS host.exemple.com;branch=z9hG4bK9320394238995
To: Recorder Application <sip:app.exemple.com>;tag=jhgf
From: Caller <sip:A@exemple.com>;tag=pqoew
Call-ID: 66676776767@app.exemple.com
CSeq: 1 NOTIFY
Max-Forwards: 70
<allOneLine>
Contact: <sip:A@exemple.com;gr=urn:uuid:f81d4fae
-7dec-11d0-a765-00a0c91e6bf6>;schemes="http,sip"
</allOneLine>
Event: refer;id=93809824
Subscription-State: active;expires=3600
Content-Type: message/sipfrag;version=2.0
Content-Length: 20

SIP/2.0 100 Trying
```

Et le serveur d'enregistrement réponds avec un 200 OK (message 10).

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS host.exemple.com;branch=z9hG4bK9320394238995
To: Recorder Application <sip:app.exemple.com>;tag=jhgf
From: Caller <sip:A@exemple.com>;tag=pqoew
Call-ID: 66676776767@app.exemple.com
CSeq: 1 NOTIFY
```

La demande REFER contenait un paramètre de champ d'en-tête Target-Dialog avec un identifiant de dialogue valide. De plus, toute la signalisation a été sur TLS et les identifiants de dialogue contiennent un aléa suffisant. À ce titre, l'appelant, A, autorise automatiquement l'application. Il agit alors sur l'URI Refer-To, allant chercher le script à app.exemple.com (message 11). La réponse, le message 12, contient une application de la Toile sur laquelle l'utilisateur peut cliquer pour activer l'enregistrement. Comme le client a exécuté l'URL dans le Refer-To, il génère un autre NOTIFY à l'application, l'informant de la réponse de succès (message 13). Il y est répondu par un 200 OK (message 14). Quand l'utilisateur clique sur le lien (message 15) les résultats sont envoyés au serveur, et une mise à jour de l'affichage est effectuée (message 16).

13. Considérations sur la sécurité

De nombreuses considérations de sécurité sont associées à ce cadre. Il permet aux applications dans le réseau d'instancier les composants d'interface d'utilisateur sur un appareil client. Ces instanciations doivent provenir d'applications authentifiées, et doivent aussi être autorisées à placer un UI dans le client. Bien sûr, l'exigence la plus forte est l'autorisation. Il n'est pas aussi important de connaître le nom du fournisseur de l'application qu'il l'est de savoir que le fournisseur est autorisé à instancier des composants.

La présente spécification définit des techniques et exigences spécifiques de l'autorisation. L'autorisation automatique est accordée si l'application peut prouver qu'elle est sur le chemin d'appel, ou qu'elle est de confiance pour un élément sur le chemin d'appel. Comme précisé ci-dessus, cela peut être accompli par l'utilisation d'identifiants de dialogue cryptographiquement aléatoires et l'usage de SIPS pour la confidentialité de message. Il est RECOMMANDÉ que SIPS soit mis en œuvre par les agents d'utilisateur conformes à la présente spécification. Cela ne représente pas de changement par rapport aux exigences de la RFC 3261.

14. Contributeurs

Le présent document a été produit par suite de discussions dans l'équipe de conception d'interaction d'application. Tous les membres de cette équipe ont contribué de façon significative aux idées incorporées dans ce document. Les membres de l'équipe étaient : Eric Burger, Cullen Jennings, Robert Fairlie-Cuninghame

15. Remerciements

Les auteurs tiennent à remercier Martin Dolly et Rohan Mahy de leurs apports et commentaires. Merci à Allison Mankin pour son soutien à ce travail.

16. Références

16.1 Références normatives

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

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

[RFC3262] J. Rosenberg et H. Schulzrinne, "Fiabilité des réponses provisoires dans le protocole d'initialisation de session (SIP)", juin 2002. *(P.S.)*

[RFC3515] R. Sparks, "Méthode Refer du protocole d'initialisation de session (SIP)", avril 2003. *(MàJ par* RFC8217)

[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

[RFC4538] J. Rosenberg, "Autorisation de demande par identification de dialogue dans le protocole d'initialisation de session (SIP)", juin 2006. *(P.S.)*

[RFC4730] E. Burger, M. Dolly, "Paquetage d'événement du protocole d'initialisation de session (SIP) pour stimulus par langage de balisage à pression de touche (KPML)", novembre 2006. *(P.S.)*

[RFC5627] J. Rosenberg, "Obtention et utilisation des URI d'agent d'utilisateur mondialement acheminable (GRUU) dans le protocole d'initialisation de session (SIP)", octobre 2009. *(P. S.)*

[Voice-XML] McGlashan, S., Lucas, B., Porter, B., Rehor, K., Burnett, D., Carter, J., Ferrans, J., and A. Hunt, "Voice Extensible Markup Language (VoiceXML) Version 2.0", W3C CR CR-voicexml20-20030220, février 2003.

16.2 Références pour information

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

[RFC3264] J. Rosenberg et H. Schulzrinne, "Modèle d'offre/réponse avec le protocole de description de session (SDP)", juin 2002. *(P.S. ; MàJ par* RFC8843, 9143)

[RFC3325] C. Jennings, J. Peterson et M. Watson, "Extensions privées au protocole d'initialisation de session (SIP) pour l'assertion d'identité au sein de réseaux de confiance", novembre 2002. *(Information ; MàJ par* RFC8217)

[RFC3550] H. Schulzrinne, S. Casner, R. Frederick et V. Jacobson, "RTP : un protocole de transport pour les applications en temps réel", STD 64, juillet 2003. *(MàJ par* RFC7164, RFC7160, RFC8083, RFC8108, RFC8860)

[RFC3680] J. Rosenberg, "Paquetage d'événements du protocole d'initialisation de session (SIP) pour les enregistrements", mars 2004. *(P.S.)*

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

[RFC4235] J. Rosenberg et autres, "Paquetage d'événement de dialogue initié par INVITE pour le protocole d'initialisation de session (SIP)", novembre 2005. *(P.S.)*

[RFC4353] J. Rosenberg, "Cadre pour les conférences avec le protocole d'initialisation de session (SIP)", février 2006. *(Information)*

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

[RFC4475] R. Sparks et autres, "Messages d'essais de résistance du protocole d'initialisation de session (SIP)", mai 2006. *(Info.)*

[RFC4733] H. Schulzrinne, T. Taylor, "Charge utile RTP pour chiffres DTMF, tonalités téléphoniques, et signaux de téléphonie", décembre 2006. *(Remplace* RFC2833) (*MàJ par* RFC4734, RFC5244) *(P.S.)*

Adresse de l'auteur

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
USA

téléphone : +1 973 952-5000
mél : jdrosen@cisco.com
URI : http://www.jdrosen.net