Groupe de travail Réseau
Request for Comments : 3880
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

J. Lennox, X. Wu, H. Schulzrinne
Columbia University
octobre 2004

Langage de traitement d'appel (CPL) : un langage pour le contrôle d'usager des services de téléphonie Internet

Statut de ce mémoire

Le présent document spécifie un protocole Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et des suggestions pour son amélioration. Prière de se reporter à l'édition actuelle du STD 1 "Normes des protocoles officiels de l'Internet" pour connaître l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé

Le présent document définit le langage de traitement d'appel (CPL, *Call Processing Language*), langage de description et de contrôle des services de téléphonie Internet. Il est conçu pour pouvoir être mis en œuvre aussi bien sur les serveurs réseau que sur les agents d'utilisateur. Il est destiné à être simple, extensible, facilement édité par les clients graphiques, et indépendant des systèmes d'exploitation ou protocoles de signalisation. Il convient à une utilisation sur un serveur où les usagers peuvent n'être pas autorisés à exécuter des programmes arbitraires, car il n'a pas de variable, de boucle, ou de capacité à faire fonctionner des programmes externes.

Table des Matières

1. Introduction

Le langage de traitement d'appel (CPL, *Call Processing Language*) est un langage qui peut être utilisé pour décrire et contrôler les services de téléphonie sur Internet. Il n'est lié à aucune architecture ou protocole de signalisation particulier ; on prévoit qu'il sera utilisé à la fois avec le protocole d'initialisation de session (SIP, *Session Initiation Protocol*) [RFC3261] et la Recommandation UIT-T [H.323].

CPL est assez puissant pour décrire un grand nombre de services et de caractéristiques, mais il est limité en puissance de sorte qu'il peut fonctionner en toute sécurité sur les serveurs de téléphonie Internet. L'intention est de rendre impossible aux usagers de faire quelque chose de plus complexe (et dangereux) que de décrire les services de téléphonie Internet. Le langage n'est pas complet au sens de Turing, et ne donne aucun moyen de créer des boucles ou des récurrences.

CPL est aussi conçu pour être facilement créé et édité par des outils graphiques. Il se fonde sur le langage de balisage extensible (XML, *Extensible Markup Language*) [XML], de sorte que son analyse est facile et que de nombreux analyseurs sont disponibles au public pour ce faire.

La structure du langage transpose étroitement son comportement, de sorte qu'un éditeur peut comprendre tout script valide, même ceux écrits à la main. Le langage est aussi conçu pour qu'un serveur puisse aisément confirmer la validité d'un script lorsque le serveur le reçoit, plutôt que de découvrir les problèmes lors du traitement d'un appel.

Les mises en œuvre de CPL sont supposées prendre place à la fois dans les serveurs de téléphonie Internet et chez les clients évolués ; tous deux peuvent utilement traiter et diriger les appels des utilisateurs. Le présent document s'adresse principalement à l'utilisation dans les serveurs. Un mécanisme sera nécessaire pour transporter les scripts entre les clients et les serveurs ; le présent document ne décrit pas un tel mécanisme, mais des documents appropriés le feront.

Le cadre et les exigences de l'architecture de CPL sont décrits dans la [RFC2824], "Cadres et exigences du langage de traitement d'appel".

1.1 Conventions du présent document

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" sont à interpréter comme décrit dans le BCP 14, RFC 2119 [RFC2119] et indiquent les niveaux d'exigence des mises en œuvre conformes de CPL.

Certains paragraphes sont en retrait pour donner les motifs et les choix de conception, des avis pour la mise en œuvre, ou des réflexions sur les futurs développements ou extensions de CPL. Ils ne sont pas essentiels à la spécification du langage, et ne sont pas normatifs.

2. Structure des scripts CPL

2.1 Structure de haut niveau

Un script CPL consiste en deux types d'informations : informations auxiliaires sur le script, et actions de traitement d'appel. Une action de traitement d'appel est une arborescence structurée qui décrit les opérations et décisions qu'effectue un serveur de signalisation téléphonique sur un événement de traitement d'appel. Il y a deux types d'actions de traitement d'appel : les actions de niveau supérieur et les sous actions. Les actions de niveau supérieur sont des actions qui sont déclenchées par des événements de signalisation qui arrivent au serveur. Deux actions de niveau supérieur sont définies : "incoming" *(entrante)*, l'action effectuée lorsque un appel arrive dont la destination est le possesseur du script, et "outgoing" *(sortante)*, l'action effectuée lorsque un appel arrive dont l'origine est le possesseur du script.

Des sous actions sont des actions qui peuvent être invoquées à partir d'autres actions. CPL interdit que de telles sous actions soient invoquées de façon récurrente : voir la Section 8.

Les information auxiliaires sont des informations qui sont nécessaires pour qu'un serveur traite correctement un script, mais qui ne décrivent directement aucune opération ou décision. Actuellement, aucune information auxiliaire n'est définie, mais la section est réservée pour être utilisée par des extensions.

2.2 Structure abstraite d'une action de traitement d'appel

Abstraitement, une action de traitement d'appel est décrite par une collection de nœuds qui décrivent des opérations qui peuvent être effectuées ou de décisions qui peuvent être prises. Un nœud peut avoir plusieurs paramètres, qui spécifient le comportement précis du nœud ; ils ont usuellement aussi des résultats, qui dépendent du résultat de la décision ou action.

Pour une représentation graphique d'une action de CPL, voir la Figure 1. Les nœuds et les résultats peuvent être vus de façon informelle comme des boîtes et des flèches ; CPL est conçu pour que les actions puissent être facilement éditées graphiquement en utilisant cette représentation. Les nœuds sont rangés dans une arborescence, partant d'un nœud à une seule racine ; les résultats des nœuds sont connectés à des nœuds supplémentaires. Lorsque une action a lieu, l'action ou décision décrite par le nœud de niveau supérieur de l'action est effectuée ; sur la base du résultat de ce nœud, le serveur suit un des résultats du nœud, et le nœud suivant sur lequel il pointe est effectué ; ce processus continue jusqu'à ce qu'un nœud sans résultat spécifié soit atteint. Parce que le graphe est acyclique, cela va se produire après que sont visités un nombre limité et prévisible de nœuds.

Si une sortie d'un nœud ne pointe pas sur un autre nœud, cela indique que le serveur CPL devrait effectuer une action spécifique d'un nœud ou du protocole. Certains nœuds ont un comportement par défaut spécifique associé ; pour d'autres, le comportement par défaut est implicite dans le protocole de signalisation sous-jacent, ou peut être configuré par l'administrateur du serveur. Voir les détails à la Section 10.

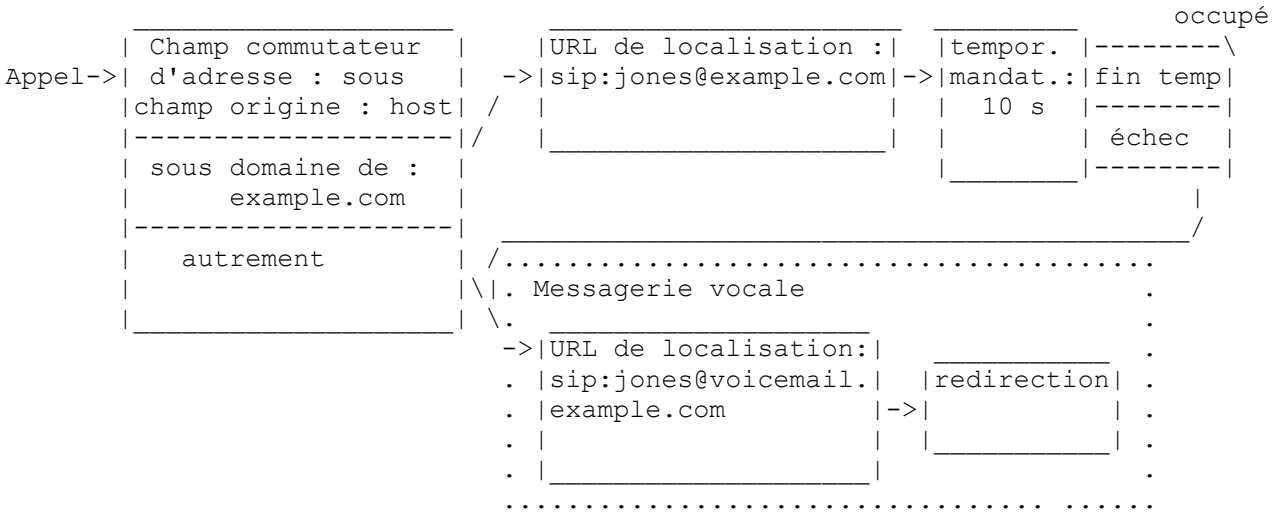

Figure 1 : Échantillon d'action CPL : version graphique

2.3 Modèle de localisation

Pour plus de souplesse, un élément d'information nécessaire pour CPL n'est pas donné comme paramètre de nœud : l'ensemble des localisations sur lesquelles un appel est à diriger. Cet ensemble de localisations est plutôt mémorisé comme une variable globale implicite tout au long de l'exécution d'une action (et ses sous actions) de traitement. Cela permet que les localisations soient restituées à partir de sources externes, filtrées, et ainsi de suite, sans exiger la prise en charge d'un langage général pour de telles opérations (qui pourrait nuire à la simplicité et la facilité de compréhension du langage). Les opérations spécifiques qui ajoutent, restituent, ou filtrent les ensembles de localisation sont données à la Section 5.

Pour l'action de niveau supérieur de traitement d'appel entrant, l'ensemble de localisation est initialisé à l'ensemble vide. Pour l'action sortante, il est initialisé à l'adresse de destination de l'appel.

2.4 Structure XML

Syntaxiquement, les scripts CPL sont représentés par des documents XML. XML est spécifié en détails par la spécification [XML], et ceux qui mettent en œuvre la présente spécification devraient être familiarisés avec ce document. Cependant, pour une brève vue d'ensemble, XML consiste en une structure hiérarchique d'étiquettes dont chacune a un certain nombre d'attributs. Il est visuellement et structurellement très similaire à HTML [HTML], car les deux langages sont des simplifications de l'ancienne, et plus large, norme SGML [ISO8879].

Voir à la Figure 2 le document XML correspondant à la représentation graphique du script CPL de la Figure 1. Les deux nœuds et les résultats en CPL sont représentés par des étiquettes XML ; les paramètres sont représentés par des attributs d'étiquette XML. Normalement, les étiquettes de nœud contiennent des étiquettes de résultat, et vice-versa (avec quelques exceptions : voir les paragraphes 5.1, 5.3, 7.1, et 7.2).

La connexion entre le résultat d'un nœud et un autre nœud est représentée en enclosant l'étiquette qui représente le nœud pointé à l'intérieur de l'étiquette pour le résultat du nœud externe. La convergence (plusieurs résultats pointant sur un seul nœud) est représentée par des sous actions, expliquées à la Section 8.

La structure de niveau supérieur d'un script CPL est représentée par des étiquettes correspondant à chaque pièce d'informations auxiliaires, de sous actions, et d'actions de niveau supérieur, dans l'ordre. Ces informations de niveau supérieur sont toutes encloses dans une étiquette spéciale "cpl", l'étiquette la plus externe du document XML.

Un schéma XML complet pour CPL est fourni à l'Appendice C. Le reste des principales sections du présent document décrit la sémantique de CPL, tout en donnant sa syntaxe de façon informelle. Pour la syntaxe formelle, se reporter à l'Appendice.

3. Structure de script : Généralités

Comme mentionné, un script CPL consiste en informations auxiliaires, sous actions, et actions de niveau supérieur. La syntaxe complète du nœud "cpl" est donnée à la Figure 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <subaction id="voicemail">
    <location url="sip:jones@voicemail.example.com">
      <redirect />
    </location>
  </subaction>
  <incoming>
    <address-switch field="origin" subfield="host">
      <address subdomain-of="example.com">
        <location url="sip:jones@example.com">
          <proxy timeout="10">
            <busy> <sub ref="voicemail" /> </busy>
```

```
        <noanswer> <sub ref="voicemail" /> </noanswer>
        <failure> <sub ref="voicemail" /> </failure>
       </proxy>
      </location>
     </address>
     <otherwise>
      <sub ref="voicemail" />
     </otherwise>
    </address-switch>
  </incoming>
</cpl>
```

Figure 2 : Échantillon de script CPL : version XML

Étiquette : "cpl"
Paramètres : aucun
Sous étiquettes : "ancillary" : voir la Section 9
"subaction" : voir la Section 8
"outgoing" : actions de niveau supérieur à effectuer sur les appels sortants de l'usager
"incoming" : actions de niveau supérieur à effectuer sur les appels entrants de l'usager

Figure 3 : Syntaxe de l'étiquette "cpl" de niveau supérieur

Les actions de traitement d'appel, actions de niveau supérieur et sous actions, consistent en une arborescence de nœuds et de résultats. Les nœuds et résultats sont tous deux décrits par des étiquettes XML. Il y a quatre catégories de nœuds CPL : les commutateurs, qui représentent les choix que peut faire un script CPL, les modificateurs de localisation, qui ajoutent ou suppriment des localisations de l'ensemble des localisations, les opérations de signalisation, qui causent des événements de signalisation dans le protocole sous-jacent, et des opérations de non signalisation, qui déclenchent un comportement qui n'affecte pas le protocole sous-jacent.

4. Commutateurs

Les commutateurs représentent les choix que peut faire un script CPL, sur la base des attributs de la demande d'appel d'origine ou des éléments indépendants de l'appel.

Tous les commutateurs sont arrangés comme des listes de conditions que peuvent satisfaire une variable. Chaque condition correspond à un résultat de nœud ; le résultat pointe sur le prochain nœud qui devrait être exécuté si la condition est vraie. Les conditions sont essayées dans l'ordre de leur présentation dans le script ; le résultat correspondant au premier nœud qui correspond est pris.

Il y a deux résultats de commutateur spéciaux qui s'appliquent à chaque type de commutateur. Le résultat "not-present", qui PEUT survenir partout dans la liste de résultats, est vrai si la variable à laquelle le commutateur était confronté n'est pas présente dans la demande d'établissement d'appel d'origine. (Dans le présent document, ceci est parfois décrit en disant que l'information est "absente".)

Le résultat "otherwise" *(autrement)*, qui DOIT être le dernier résultat spécifié si il est présent, correspond si aucune autre condition n'est satisfaite.

Si aucune condition ne correspond et si il n'y a pas de résultat "otherwise" présent dans le script, le comportement de script par défaut est pris. Voir à la Section 10 plus d'informations sur cette question.

Les commutateurs PEUVENT ne contenir aucun résultat. Ils PEUVENT ne contenir qu'un résultat "otherwise".

De tels commutateurs ne sont pas particulièrement utiles, mais peuvent être créés par des outils qui génèrent automatiquement des scripts CPL.

4.1 Commutateurs d'adresses

Les commutateurs d'adresses permettent à un script CPL de prendre des décisions sur la base d'une des adresses présentes dans la demande d'appel d'origine. Ils sont résumés dans la Figure 4.

Nœud : "address-switch"
Résultats : "address" adresses spécifiques à satisfaire
Paramètres : "field" "origin", "destination", ou "original-destination"
"subfield" "address-type", "user", "host", "port", "tel", ou "display" (aussi : "password" et "alias-type")
Résultat : "address"
Paramètres : "is" correspondance exacte
"contains" correspondance de sous chaîne (seulement pour "display")
"subdomain-of" correspondance de sous domaine (pour "host", "tel")

Figure 4 : Syntaxe du nœud "address-switch"

Les commutateurs d'adresses ont deux paramètres de nœud : "field" et "subfield". Le paramètre obligatoire "field" *(champ)* permet au script de spécifier quelle adresse est à considérer pour le commutateur : soit l'adresse d'origine de l'appel (champ "origin") son adresse de destination actuelle (champ "destination") soit son champ de destination d'origine (champ "original-destination") la destination que l'appel avait avant l'invocation de toute transmission antérieure. Les serveurs PEUVENT définir des valeurs de champ supplémentaires.

Le "subfield" *(sous champ)* facultatif spécifie quelle partie de l'adresse est à considérer. Les valeurs de sous champ possibles sont : "address-type", "user", "host", "port", "tel", et "display". Des valeurs supplémentaires de sous champ PEUVENT être définies pour des valeurs spécifiques du protocole. (Le sous champ "password" est défini pour SIP au paragraphe 4.1.1 ; le sous champ "alias-type" est défini pour H.323 à l'Appendice B.1.) Si aucun sous champ n'est spécifié, cela correspond à l'adresse "entière" ; la signification précise est définie pour chaque protocole de signalisation sous-jacent. Les serveurs PEUVENT définir des valeurs de sous champ supplémentaires.

Les sous champs sont définis comme suit :

address-type : cela indique le type de l'adresse sous-jacente, c'est-à-dire, le schéma d'URI, si l'adresse peut être représentée par un URI. Les types spécifiquement discutés dans le présent document sont "sip", "tel", et "h323". Le type d'adresse est insensible à la casse. Il a une valeur pour tous les types d'adresse définis.

user : ce sous champ de l'adresse indique, pour les adresses de style messagerie électronique la partie usager de l'adresse. Pour une adresse de style numéro de téléphone, il comporte le numéro d'abonné. Ce sous champ est sensible à la casse ; il peut être absent.

host : ce sous champ de l'adresse indique le nom d'hôte Internet ou l'adresse IP correspondant à l'adresse, en format de représentation textuelle de nom d'hôte, IPv4, ou IPv6 [RFC3513]. Les noms d'hôte sont comparés en tant que chaînes. Les adresses IP sont comparées numériquement. (En particulier, la présence ou la localisation d'un bloc IPv6 :: de bits zéro omis n'est pas significative pour la confrontation.) Les noms d'hôte ne sont jamais égaux aux adresses IP – aucune résolution DNS n'est effectuée. Les adresses IPv4 ne sont jamais égales aux adresses IPv6, même si l'adresse IPv6 est une incorporation v4-dans-v6. Ce sous champ n'est pas sensible à la casse, et peut être absent.

Pour les seuls noms d'hôte, la confrontation de sous domaine est prise en charge par l'opérateur de correspondance "subdomain-of" *(sous domaine de)*. L'opérateur "subdomain-of" ignore les points en tête du nom d'hôte ou du schéma de correspondance, s'il en est.

port : ce sous champ indique le numéro d'accès TCP ou UDP de l'adresse, numériquement, en format décimal. Il n'est pas sensible à la casse, car il DOIT seulement contenir des chiffres décimaux. Les zéros en tête sont ignorés.

tel : ce sous champ indique un numéro de téléphone d'abonné, si l'adresse contient un tel numéro. Il n'est pas sensible à la casse (les numéros de téléphone peuvent contenir les symboles 'A', 'B', 'C', ou 'D') et peut être absent. Il peut être confronté en utilisant l'opérateur de confrontation "subdomain-of". Les caractères de ponctuation et de séparation dans les numéros de téléphone sont éliminés.

display : ce sous champ indique un "nom d'affichage" ou un nom visible par l'utilisateur correspondant à une adresse. C'est une chaîne Unicode, et la confrontation se fait en utilisant un algorithme insensible à la casse décrit au paragraphe 4.2. L'opérateur "contains" peut lui être appliqué. Il peut être absent.

Pour tout sous champ complètement inconnu, le serveur PEUT rejeter le script au moment de sa soumission avec une indication du problème ; si un script avec un sous champ inconnu est exécuté, le serveur DOIT considérer le résultat "not-present" comme étant le résultat valide.

L'étiquette de résultat "address" peut prendre exactement un des trois paramètres possibles suivants, qui indiquent la sorte de correspondance permise.

is : un résultat avec cet opérateur de correspondance est suivi si le sous champ qui est confronté dans le "address-switch" correspond exactement à l'argument de l'opérateur. Il peut être utilisé pour tout sous champ, ou pour l'adresse entière si aucun sous champ n'était spécifié.

subdomain-of : cet opérateur de correspondance s'applique seulement pour les sous champs "host" et "tel". Dans le premier cas, il correspond si le nom d'hôte confronté est un sous domaine du domaine donné dans l'argument de l'opérateur de correspondance ; donc, subdomain-of="example.com" va correspondre aux noms d'hôtes "example.com", "research.example.com", et "zaphod.sales.internal.example.com". Des adresses IP peuvent être données comme arguments de cet opérateur ; cependant, elles ne correspondent qu'exactement. Dans le cas du sous champ "tel", le résultat correspond si le numéro de téléphone confronté a un préfixe qui correspond à l'argument de l'opérateur de correspondance ; le subdomain-of="1212555" va correspondre au numéro de téléphone "1 212 555 1212."

contains : cet opérateur de correspondance s'applique seulement pour le sous champ "display". Le résultat correspond si le nom affiché confronté contient l'argument de la confrontation comme sous chaîne.

4.1.1 Usage de "address-switch" avec SIP

Pour SIP, l'adresse "origin" correspond à l'adresse dans l'en-tête "From", "destination" correspond à l'en-tête "Request-URI", et "original-destination" correspond à l'en-tête "To".

Le sous champ "display" d'une adresse est la partie nom d'affichage de l'adresse, si elle est présente. À cause de la syntaxe de SIP, le champ d'adresse "destination" ne va jamais avoir un sous champ "display".

Le sous champ "address-type" d'une adresse est le schéma d'URI de cette adresse. D'autres champs d'adresse dépendent de ce "address-type".

Pour les URI SIP, les sous champs "user", "host", et "port" correspondent aux éléments "user," "host," et "port" de la syntaxe d'URI. (Noter que, suivant les définitions de la [RFC3261], un URI SIP qui ne spécifie pas un accès n'est pas le même qu'un accès 5060 explicite ; le premier est indiqué par un sous champ port absent.) Le sous champ "tel" est défini comme étant la partie "user" de l'URI, dont les séparateurs visuels sont supprimés, si le paramètre "user=phone" est donné à l'URI, ou si le serveur est par ailleurs configuré à reconnaître la partie usager comme un numéro de téléphone. Un sous champ supplémentaire, "password", est défini comme correspondant à l'élément "password" de l'URI SIP, et il est sensible à la casse. Cependant, l'utilisation de ce champ N'EST PAS RECOMMANDÉE pour des raisons générales de sécurité.

Pour les URI tel, les sous champs "tel" et "user" sont le nom de l'abonné ; dans le premier cas, les séparateurs visuels sont supprimés. Les sous champs "host" et "port" sont tous deux absents.

Pour les URL h323, les sous champs PEUVENT être réglés selon le schéma décrit à l'Appendice B.

Pour les autres schémas d'URI, seul le sous champ "address-type" est défini par la présente spécification ; les serveurs PEUVENT établir d'autres sous champs prédéfinis, ou PEUVENT prendre en charge des sous champs supplémentaires.

Si aucun sous champ n'est spécifié pour les adresses dans les messages SIP, la chaîne confrontée est la partie URI de l'adresse. Pour les confrontations de "is", les règles standard de correspondance d'URI SIP sont utilisées ; pour les confrontations à "contains", l'URI est utilisé tel quel.

4.2 Commutateurs de chaînes

Les commutateurs de chaînes permettent à un script CPL de prendre des décisions sur la base de chaînes de forme libre présentes dans une demande d'appel. Elles sont résumées à la Figure 5.

Nœud : "string-switch"
Résultats : "string" chaîne spécifique à satisfaire
Paramètres : "field" "subject", "organization", "user-agent", ou "display"

Résultat : "string"
Paramètres : "is" correspondance exacte
contains" correspondance de sous chaîne

Figure 5 : Syntaxe du nœud "string-switch"

Les commutateurs de chaînes ont un paramètre de nœud : "field". Le paramètre obligatoire "field" spécifie quelle chaîne est à satisfaire.

Les commutateurs de chaînes dépendent des protocoles de signalisation d'appel utilisés.

Quatre champs sont définis qui figurent ci-dessous. La valeur de chacun de ces champs est une chaîne Unicode de forme libre sans autre structure définie.
subject : l'objet de l'appel.
organization : l'organisation de l'origine de l'appel.
user-agent : le nom du programme ou appareil avec lequel est faite la demande d'appel.
display : un texte de forme libre associé à l'appel, destiné à être affiché au receveur, sans autre sémantique définie par le protocole de signalisation.

Les chaînes sont confrontées comme des chaînes Unicode insensibles à la casse, de la manière suivante. D'abord, les chaînes sont mises en forme canonique de composition de compatibilité (KC, *Compatibility Composition*) comme spécifié dans la norme Unicode Annexe 15 [UNICODE]. Ensuite, les chaînes sont comparées en utilisant la transposition insensible à la casse locale, comme spécifié dans l'Annexe 21 de la norme Unicode [TRANSP].

Le code pour effectuer la première étape, en Java et Perl, est disponible ; voir les liens de l'Annexe 5 de UAX 15 [UNICODE]. La comparaison de chaîne insensible à la casse dans les bibliothèques de classe standard de Java effectue déjà la seconde étape ; les autres bibliothèques à capacité Unicode devraient être similaires.

L'étiquette de résultat de la confrontation de chaîne est nommée "string", et a un argument obligatoire, soit "is", soit "contains", indiquant respectivement une correspondance de chaîne entière ou de sous chaîne.

4.2.1 Usage de "string-switch" avec SIP

Pour SIP, les champs "subject", "organization", et "user-agent" correspondent aux champs d'en-tête SIP avec le même nom. Ils sont utilisés tels qu'ils apparaissent dans le message.

Le champ "display" n'est pas utilisé, et n'est jamais présent.

4.3 Commutateurs de langage

Les commutateurs de langage permettent à un script CPL de prendre des décisions sur la base des langages dans lesquels le générateur de l'appel souhaite communiquer. Ils sont résumés à la Figure 6.

Nœud : "language-switch"
Résultats : "language" chaîne spécifique à satisfaire
Paramètres : aucun

Résultat : "language"
Paramètres : "matches" correspond si le langage mentionné correspond à une gamme de langues de l'appel

Figure 6: Syntaxe du nœud "language-switch"

Les commutateurs de langage ne prennent pas de paramètre.
Le résultat "language" prend un paramètre, "matches". La valeur du paramètre est une étiquette de langage, comme défini dans la [RFC3066]. L'appelant peut avoir spécifié un ensemble de gammes de langages, aussi comme défini dans la RFC3066. Le serveur CPL vérifie chaque étiquette de langue spécifiée par le script par rapport aux gammes de langues spécifiées dans la demande. Voir dans la RFC 3066 les détails de la façon dont les gammes de langues correspondent aux étiquettes de langage. En bref, une gamme de langage correspond à une étiquette de langue si elle est exactement égale à l'étiquette, ou si elle est exactement égale à un préfixe de l'étiquette comme lorsque le premier caractère qui suit le préfixe est "-".

Si l'appelant a spécifié la gamme de langage spéciale "*", elle est ignorée pour les besoins de la confrontation. Les langages qui ont une valeur "q" de 0 sont aussi ignorés.

Ce commutateur PEUT être absent.

4.3.1 Usage de "language-switch" avec SIP

Les gammes de langues pour le commutateur "language-switch" sont obtenues du champ d'en-tête SIP "Accept-Language". Le commutateur n'est pas présent si la demande initiale SIP ne contenait pas ce champ d'en-tête.

Noter que à cause de la sémantique de première correspondance de CPL dans les commutateurs, les valeurs "q" autres que 0 des champs d'en-tête "Accept-Language" sont ignorées.

4.4 Commutateurs horaires

Les commutateurs horaires permettent à un script CPL de prendre des décisions sur la base de l'heure et/ou date à laquelle le script est exécuté. Ils sont récapitulés à la Figure 7.

Les commutateurs horaires sont indépendants du protocole de signalisation sous-jacent.

Nœud : "time-switch"
Résultats : "time" L'heure spécifique à satisfaire.
Paramètres : "tzid" Identifiant de zone horaire de la RFC 2445.
"tzurl" URL de zone horaire de la RFC 2445.

Résultat : "time"
Paramètres :
"dtstart" Début de l'intervalle (DATE-HEURE de la RFC 2445)
"dtend" Fin de l'intervalle (DATE-HEURE de la RFC 2445)
"duration" Longueur de l'intervalle (DURATION de la RFC 2445)
"freq" Fréquence de récurrence ("secondly", "minutely", "hourly", "daily", "weekly", "monthly", ou "yearly")
"interval" Intervalle de répétition de la récurrence.
"until" Limites de la récurrence (DATE-HEURE de la RFC 2445)
"count" Nombre d'occurrences de la récurrence.
"bysecond" Liste de secondes dans une minute.
"byminute" Liste de minutes dans une heure.
"byhour" Liste des heures de la journée.
"byday" Liste des jours dans la semaine.
"bymonthday" Liste des jours dans le mois.
"byyearday" Liste des jours dans l'année.
"byweekno" Liste des semaines dans l'année.
"bymonth" Liste des mois dans l'année.
"wkst" Premier jour ouvré de la semaine.
"bysetpos" Liste des valeurs dans l'ensemble des événements spécifiés.

Figure 7 : Syntaxe du nœud "time-switch"

Les commutateurs horaires se fondent étroitement sur la spécification d'intervalles de temps récurrents de la spécification centrale des objets de calendrier et de programmation de l'Internet (iCalendar COS), [RFC2445].

Cela permet aux scripts CPL d'être générés automatiquement à partir de livres calendaires. Cela permet aussi de réutiliser l'immense travail existant de spécification des intervalles de temps.

Si de futurs documents sur la voie de la normalisation étaient publiés pour mettre à jour ou remplacer la RFC 2445, tout changement ou précision que ces documents apporteraient au traitement de la récurrence s'appliquera aussi aux commutateurs horaires de CPL.

L'Appendice A donne un algorithme pour déterminer si un instant entre dans une certaine récurrence.

L'étiquette "time-switch" prend deux paramètres facultatifs, "tzid" et "tzurl", qui sont tous deux définis dans la RFC 2445 (respectivement aux paragraphes 4.8.3.1 et 4.8.3.5). L'étiquette "tzid" identifie la définition de zone horaire référencée. Si elle commence par une barre oblique (solidus), elle fait référence à un registre de zone horaire mondiale à définir ; autrement, elle est définie en local au serveur. L'étiquette "tzurl" donne une localisation de réseau à partir de laquelle une définition VTIMEZONE à jour peut être retrouvée pour la zone horaire.

Bien que les étiquettes "tzid" qui ne commencent pas par une barre oblique soient définies en local, il est RECOMMANDÉ que les serveurs prennent en charge au moins le schéma de dénomination utilisé par la base de données Olson des zones horaires [Sources]. Des exemples de bases de données de zone horaire qui utilisent le schéma Olson sont les fichiers

zoneinfo sur la plupart des systèmes de style Unix, et la classe de standard Java TimeZone.

Les serveurs DEVRAIENT résoudre les références "tzid" et "tzurl" aux définitions de zone horaire au moment du chargement du script. Ils PEUVENT rafraîchir périodiquement ces résolutions pour obtenir la définition la plus à jour de zone horaire. Si une "tzurl" devient invalide, les serveurs DEVRAIENT mémoriser les données valides les plus récentes récupérées de l'URL.

Si un script est chargé avec des "tzid" et "tzurl" que le serveur CPL ne reconnaît pas ou ne peut pas résoudre, il DEVRAIT le diagnostiquer et les rejeter au moment du chargement du script. Si ni "tzid" ni "tzurl" ne sont présentes, toutes les heures non UTC dans ce commutateur horaire devraient être interprétées comme étant des heures "flottantes", c'est-à-dire, qu'elles sont spécifiées dans la zone horaire locale du serveur CPL.

Comme l'heure d'été change au fil de l'an, il est nécessaire de spécifier les commutations horaires dans une certaine zone horaire. Les décalages de l'UTC ne sont pas suffisants, car une règle d'acheminement à une heure de la journée qui est fixée entre 9 heure et 17 heure dans l'Est des USA va commencer entre 8 heure et 16 heure à la fin octobre.

Les auteurs de serveurs CPL devraient veiller à traiter correctement les intervalles lorsque l'heure locale est discontinue au début ou à la fin d'une période d'heure d'été. Noter en particulier que certains horaires peuvent survenir plusieurs fois lorsque les horloges sont reculées. L'algorithme de l'Appendice A est supposé traiter cela correctement.

Les nœuds horaires spécifient une liste de périodes durant lesquelles leurs résultats devraient être retenus. Ils ont deux paramètres obligatoires : "dtstart", qui spécifie le début de la première période de la liste, et exactement un de "dtend" ou "duration", qui spécifient respectivement l'heure de fin ou la durée de la période. Les paramètres "dtstart" et "dtend" sont formatés comme des valeurs iCalendar COS DATE-TIME, comme spécifié au paragraphe 4.3.5 de la [RFC2445]. Comme les zones horaires sont spécifiées dans l'étiquette de niveau supérieur "time-switch", seules les formes 1 ou 2 (heure flottante ou UTC) peuvent être utilisées. Le paramètre "duration" est donné comme paramètre iCalendar COS DURATION, comme spécifié au paragraphe 4.3.6 de la RFC2445. Les syntaxes DATE-TIME et DURATION sont toutes deux des sous ensembles des syntaxes correspondantes de [ISO8601].

Pour un intervalle récurent, le paramètre "duration" DOIT être assez petit pour que les intervalles suivants ne se chevauchent pas. Pour les intervalles non récurrents, des durées de toutes longueurs positives sont permises. Les durées de longueur zéro et négatives ne sont pas permises.

Si aucun autre paramètre n'est spécifié, un nœud horaire indique seulement une période horaire. Des ensembles plus compliqués d'intervalles horaires sont construits comme des récurrences. Une récurrence est spécifiée en incluant le paramètre "freq", qui indique le type de règle de récurrence. Des paramètres autres que "dtstart", "dtend", et "duration" NE DEVRAIENT PAS être spécifiés sauf si "freq" est présent, mais les serveurs CPL DEVRAIENT accepter les scripts avec la présence de tels paramètres, et ignorer les autres paramètres.

Le paramètre "freq" prend une des valeurs suivantes : "secondly" *(par seconde)*, pour spécifier des périodes répétées fondées sur un intervalle d'une seconde ou plus, "minutely" *(par minute)*, pour spécifier des périodes répétées sur la base d'un intervalle d'une minute ou plus, "hourly" *(par heure)*, pour spécifier des périodes répétées sur la base d'un intervalle d'une heure ou plus, "daily" *(quotidiennement)*, pour spécifier des périodes répétées sur la base d'un intervalle d'un jour ou plus, "weekly" *(hedomadaire)*, pour spécifier des périodes répétées sur la base d'un intervalle d'une semaine ou plus, "monthly" *(mensuellement)*, pour spécifier des périodes répétées sur la base d'un intervalle d'un mois ou plus, et "yearly" *(annuellement)*, pour spécifier des périodes répétées sur la base d'un intervalle d'une année ou plus. Ces valeurs ne sont pas sensibles à la casse.

Le paramètre "interval" contient un entier positif qui représente combien de fois se répète la règle de récurrence. La valeur par défaut est "1", qui signifie chaque seconde pour une règle "secondly", chaque minute pour une règle "minutely", chaque heure pour une règle "hourly", chaque jour pour une règle "daily", chaque semaine pour une règle "weekly", chaque mois pour une règle "monthly", et chaque année pour une règle "yearly".

Le paramètre "until" définit une valeur COS DATE ou DATE-TIME de iCalendar qui borde la règle de récurrence de façon inclusive. Si la valeur spécifiée par "until" est synchronisée avec la récurrence spécifiée, cette date ou date-heure devient la dernière instance de la récurrence. Si il est spécifié comme une valeur de date-heure, il DOIT alors être spécifié en format d'heure UTC. Si il n'est pas présent, et si le paramètre "count" n'est pas présent non plus, la récurrence est considérée se répéter à l'infini.

Le paramètre "count" définit le nombre d'occurrences qui forme la gamme limite de la récurrence. Le paramètre "dtstart" compte comme première occurrence. Les paramètres "until" et "count" NE DOIVENT PAS se produire dans le même "time".

Le paramètre "bysecond" spécifie une liste de secondes séparées par des virgules dans une minute. Les valeurs valides sont de 0 à 59. Le paramètre "byminute" spécifie une liste de minutes séparées par des virgules dans une heure. Les valeurs valides sont de 0 à 59. Le paramètre "byhour" spécifie une liste d'heures séparées par des virgules dans une journée. Les valeurs valides sont de 0 à 23.

Le paramètre "byday" spécifie une liste séparée par des virgules de jours de la semaine. "MO" indique lundi, "TU" indique mardi, "WE" indique mercredi, "TH" indique jeudi, "FR" indique vendredi, "SA" indique samedi, et "SU" indique dimanche. Ces valeurs ne sont pas sensibles à la casse.

Chaque valeur "byday" peut aussi être précédée d'un entier positif (+n) ou négatif (-n). Si c'est présent, cela indique la nième occurrence du jour spécifique au sein de la récurrence "monthly" ou "yearly". Par exemple, au sein d'une règle "monthly", +1MO (ou simplement 1MO) représente le premier lundi du mois, tandis que -1MO représente le dernier lundi du mois. Si un modificateur entier n'est pas présent, cela signifie tous les jours de ce type dans la fréquence spécifiée. Par exemple, dans une règle "monthly", MO représente tous les lundis dans le mois.

Le paramètre "bymonthday" spécifie une liste séparée par des virgules des jours du mois. Les valeurs valides sont de 1 à 31 ou de -31 à -1. Par exemple, -10 représente du dixième au dernier jour du mois.

Le paramètre "byyearday" spécifie une liste séparée par des virgules des jours de l'année. Les valeurs valides sont de 1 à 366 ou de -366 à -1. Par exemple, -1 représente le dernier jour de l'année (le 31 décembre) et -306 représente du 306 ème (1er novembre) au dernier jour de l'année.

Le paramètre "byweekno" spécifie une liste séparée par des virgules d'ordinaux qui spécifient des semaines dans l'année. Les valeurs valides sont de 1 à 53 ou de -53 à -1. Cela correspond aux semaines conformément à la numérotation des semaines définie dans la norme ISO 8601 [ISO8601]. Une semaine est définie comme une période de sept jours, commençant au jour de la semaine défini par le début de semaine (voir "wkst"). La semaine numéro un de l'année calendaire est la première semaine qui contient au moins quatre (4) jours dans cette année calendaire. Ce paramètre n'est valide que pour les règles "yearly". Par exemple, 3 représente la troisième semaine de l'année.

Note : En supposant un début de semaine le lundi, la semaine 53 ne peut survenir que lorsque le premier janvier est un jeudi ou, pour les années bissextiles, si le 1er janvier est un mercredi.

Le paramètre "bymonth" spécifie une liste séparée par des virgules des mois de l'année. Les valeurs valides sont de 1 à 12.

Le paramètre "wkst" spécifie le jour de début de la semaine de travail. Les valeurs valides ont "MO", "TU", "WE", "TH", "FR", "SA" et "SU". Ceci est significatif lorsque une récurrence "weekly" a un intervalle supérieur à 1, et qu'un paramètre "byday" est spécifié. C'est aussi significatif dans une récurrence "yearly" lorsque un paramètre "byweekno" est spécifié. La valeur par défaut est "MO", conformément à la norme ISO 8601 [ISO8601].

Le paramètre "bysetpos" spécifie une liste séparée par des virgules des valeurs qui correspondent à la nième occurrence dans l'ensemble des événements spécifiés par la règle. Les valeurs valides sont de 1 à 366 ou de -366 à -1. Il ne DOIT être utilisé qu'en conjonction avec un autre paramètre byxxx. Par exemple, "le dernier jour ouvré du mois" pourrait être représenté comme : <time -timerange- freq="monthly" byday="MO,TU,WE,TH,FR" bysetpos="-1">

Chaque valeur "bysetpos" peut inclure un entier positif (+n) ou négatif (-n). Si il est présent, cela indique la nième occurrence de l'occurrence spécifique au sein de l'ensemble des événements spécifiés par la règle.

Si des valeurs de paramètre byxxx se trouvent au delà de la portée disponible (c'est-à-dire, bymonthday="30" en février) elles sont simplement ignorées.

Les paramètres byxxx modifient d'une certaine manière la récurrence. La règle byxxx qui partage une période supérieure ou égale à la fréquence réduit généralement ou limite le nombre d'occurrences de la récurrence générée. Par exemple, freq="daily" bymonth="1" réduit le nombre d'instances de récurrence de tous les jours (si le paramètre "bymonth" n'est pas présent) à tous les jours de janvier. Les paramètres byxxx pour une période inférieure à la fréquence augmentent ou étendent généralement le nombre d'occurrences de la récurrence. Par exemple, freq="yearly" bymonth="1,2" augmente le nombre de jours au sein de l'ensemble de récurrence annuelle de 1 (si le paramètre "bymonth" n'est pas présent) à 2.

Si plusieurs paramètres byxxx sont spécifiés, alors après avoir évalué les paramètres "freq" et "interval" spécifiés, les paramètres byxxx sont appliqués à l'ensemble actuel d'occurrences évaluées dans l'ordre suivant : "bymonth", "byweekno", "byyearday", "bymonthday", "byday", "byhour", "byminute", "bysecond", et "bysetpos" ; puis sont évaluées "count" et "until".

Voici un exemple d'évaluation de paramètres byxxx multiples.

<time dtstart="19970105T083000" duration="10M"
freq="yearly" interval="2" bymonth="1" byday="SU"
byhour="8,9" byminute="30">

D'abord, interval="2" sera appliqué à freq="yearly" pour arriver à "tous les deux ans". Puis, bymonth="1" va s'appliquer pour arriver à "tous les janviers, tous les deux ans". Puis, byday="SU" va être appliqué pour arriver à "tous les dimanches de janvier, tous les deux ans". Puis, byhour="8,9" va s'appliquer pour arriver à "tous les dimanches de janvier à 8 h et 9 h, tous les deux ans". Ensuite on va appliquer byminute="30" pour arriver à "tous les dimanches de janvier à 8:30 et 9:30, tous les deux ans". Puis les secondes sont déduites de "dtstart" pour finir par "tous les dimanches de janvier de 8:30:00 à 8:40:00, et de 9:30:00 à 9:40:00, tous les deux ans". De même, si le paramètre "byminute", "byhour", "byday", "bymonthday", ou "bymonth" manque, la minute, heure, journée, ou mois approprié serait restituée à partir du paramètre "dtstart".

Les règles de récurrence iCalendar COS RDATE, EXRULE, et EXDATE ne sont pas spécifiquement transposées en composants du nœud de commutation horaire. On peut obtenir une fonctionnalité équivalente aux règles d'exception en utilisant l'ordre des règles de commutation pour exclure des périodes avec des règles antérieures ; une fonctionnalité équivalente aux règles RDATE de date supplémentaire peut être atteinte en utilisant des nœuds "sub" (voir la Section 8) pour lier plusieurs résultats au même nœud suivant.

Le résultat "not-present" n'est jamais vrai pour un commutateur horaire. Cependant, il PEUT être inclus pour permettre un traitement plus régulier du commutateur.

4.4.1 Différences de iCalendar et questions de mise en œuvre

(Ce paragraphe n'est pas normatif.)

La spécification des événements récurrents dans ce paragraphe est identique (sauf les questions de syntaxe et de formatage) à celle de la [RFC2445], avec une seule restriction supplémentaire. Cette restriction est que les instances consécutives d'intervalles de récurrence ne doivent pas se chevaucher.

Cela a donné lieu des débats, durant la conception du CPL, pour savoir si la spécification entière de la récurrence de iCalendar COS devrait être incluse dans le CPL, ou si seulement un sous ensemble devrait être inclus. Il a été finalement décidé que la compatibilité entre les deux protocoles était de première importance. Cela impose des questions supplémentaires de mise en œuvre pour les serveurs CPL.

Il apparaît qu'il n'est pas possible de déterminer, en temps constant, si un certain instant tombe dans un des intervalles définis par une pleine récurrence de iCalendar COS. Les principaux soucis sont :

- Le paramètre "count" ne peut pas être vérifié en temps constant courant, car il exige qu'un serveur énumère toutes les récurrences à partir de "dtstart" jusqu'à l'instant présent, afin de déterminer si la récurrence courante satisfait le paramètre. Cependant, un serveur peut étendre une fois un paramètre "count", hors ligne, pour déterminer la date de la dernière récurrence. Cette date peut alors être traitée comme un paramètre "until" virtuel pour le traitement interne du serveur.

- De même, le paramètre "bysetpos" exige qu'un serveur énumère toutes les instances de l'occurrence depuis le début de l'ensemble courant de récurrences jusqu'à l'instant présent. Cela exige un prétraitement un peu plus complexe, mais généralement, une seule récurrence avec un paramètre "bysetpos" peut être partagé en plusieurs récurrences sans lui.

- Finalement, l'écoulement constant du temps des commutateurs horaires exige aussi qu'on puisse établir une heure de début candidate pour une récurrence de façon rapide et univoque, pour vérifier si elle satisfait aux autres restrictions. Cela impose que la durée d'une récurrence ne soit pas plus longue que son intervalle de répétition, afin qu'un instant donné ne puisse pas tomber entre plusieurs répétitions potentiellement consécutives de la récurrence. La restriction que des intervalles consécutifs ne se chevauchent pas partiellement satisfait à cette condition, mais ne la garantit pas complètement. Là encore, le prétraitement peut dans une certaine mesure aider à résoudre le problème.

L'algorithme de l'Appendice A fonctionne en temps constant après ces étapes de pré traitement.

Les serveurs devraient vérifier que les règles de récurrence ne créent pas d'exigences absurdes en matière de démarrage ou de mémoire, et rejeter celles qui le font, tout comme ils devraient vérifier que les scripts CPL en général ne sont pas d'une taille démesurée.

4.5 Commutateurs de priorité

Les commutateurs de priorité permettent à un script CPL de prendre des décisions sur la base de la priorité spécifiée pour l'appel original. Ils sont récapitulés à la Figure 8. Ils dépendent du protocole de signalisation sous-jacent .

Nœud : "priority-switch"
Résultats : "priority" La priorité spécifique à satisfaire
Paramètres : aucun

Résultat : "priority"
Paramètres : "less" *(moins)* correspond si la priorité est moins que ce qui est spécifié.
"greater" *(supérieur à)* correspond si la priorité est plus que ce qui est spécifié.
"equal" *(égal)* correspond si la priorité est égale à ce qui est spécifié.

Figure 8: Syntaxe du nœud "priority-switch"

Les commutateurs de priorité ne prennent pas de paramètre.

L'étiquette "priority" prend un des trois paramètres "greater", "less", ou "equal". Les valeurs de ces paramètres sont une des priorités suivantes en ordre décroissant : "emergency", "urgent", "normal", et "non-urgent". Ces valeurs sont confrontées de façon insensible à la casse. Les résultats avec le paramètre "less" sont pris si la priorité de l'appel est moins que la priorité donnée dans l'argument, et ainsi de suite.

Si aucune priorité n'est spécifiée dans un message, la priorité est considérée comme étant "normal". Si une priorité inconnue est spécifiée dans l'appel, elle est considérée comme équivalente à "normal" pour les besoins de comparaison avec "greater" et "less", mais elle est comparée littéralement pour la comparaison avec "equal".

Comme tout message a une priorité, le résultat "not-present" n'est jamais vrai pour un commutateur de priorité. Cependant, il PEUT être inclus, pour permettre un traitement plus régulier du traitement de commutateur.

4.5.1 Usage de "priority-switch" avec SIP

La priorité d'un message SIP correspond à l'en-tête "Priority" dans le message "INVITE" initial.

5. Modificateurs de localisation

Le modèle de localisation abstrait de CPL est décrit au paragraphe 2.3. Le comportement de plusieurs des opérations de signalisation (définies à la Section 6) dépend de l'ensemble de localisation spécifié. Les nœuds de localisation ajoutent ou suppriment des localisations de l'ensemble des localisations.

Trois types de nœuds de localisation sont définis. Les localisations explicites ajoutent des localisations spécifiées de façon littérale à l'ensemble de localisations courant, les recherches de localisations obtiennent des localisations d'une source externe, et les filtres de localisation suppriment des localisations de l'ensemble, sur la base de critères spécifiés.

5.1 Localisation explicite

Les nœuds de localisation explicite spécifient littéralement une localisation. Leur syntaxe est décrite à la Figure 9.

Les nœuds de localisation explicite dépendent du protocole de signalisation sous-jacent.

Nœud : "location"
Résultats : aucun (le prochain nœud suit directement)
Prochain nœud : tout nœud
Paramètres : "url" URL de l'adresse à ajouter à l'ensemble de localisation
"priority" Priorité de cette localisation (0,0 à 1,0)
"clear" si il faut purger l'ensemble de localisation avant d'ajouter la nouvelle valeur

Figure 9: Syntaxe du nœud "localisation"

Les nœuds de localisation explicite ont trois paramètres de nœud. La valeur du paramètre "url" obligatoire est l'URL de l'adresse à ajouter à l'ensemble de localisations. Une seule adresse peut être spécifiée par nœud de localisation ; plusieurs

localisations peuvent être spécifiées en mettant ces nœuds en cascade.

Le paramètre facultatif "priority" spécifie une priorité pour la localisation. Sa valeur est un nombre à virgule flottante entre 0,0 et 1,0. Si il n'est pas spécifié, le serveur DEVRAIT supposer une priorité par défaut de 1,0. Le paramètre facultatif "clear" spécifie si l'ensemble de localisation devrait être purgé avant d'y ajouter la nouvelle localisation. Sa valeur peut être "yes" ou "no", avec "no" par défaut.

Les nœuds de localisation de base ont seulement un résultat possible, car il n'y a rien qui puisse les faire échouer. (Si un nœud de localisation de base spécifie une localisation qui n'est pas prise en charge par le protocole de signalisation sous-jacent, le serveur de script DEVRAIT le détecter et le rapporter à l'usager au moment de la soumission du script.) Donc, leurs représentations XML n'ont pas d'étiquette de résultat explicite ; l'étiquette <localisation> contient directement un autre nœud.

5.1.1 Usage de "localisation" avec SIP

Toutes les localisations SIP sont représentées comme des URL, de sorte que les localisations spécifiées dans les étiquettes "location" sont interprétées directement.

5.2 Recherche de localisation

Les localisations peuvent aussi être spécifiées par des moyens externes, en utilisant des recherches de localisation. La syntaxe de ces étiquettes est donnée à la Figure 10.

La recherche de localisation dépend du protocole de signalisation sous-jacent.

```
Nœud :  "lookup"
Résultats :  "success"     Prochain nœud si la recherche a réussi.
             "notfound"  Prochain nœud si la recherche n'a pas trouvé d'adresse
             "failure"     Prochain nœud si la recherche a échoué.
Paramètres :  "source"    Source de la recherche.
              "timeout"   Durée des essais avant d'abandonner la recherche.
              "clear"      Si il faut purger l'ensemble de localisations avant d'ajouter les nouvelles valeurs.
Résultat :  "success"
Paramètres : aucun
Résultat :  "notfound"
Paramètres : aucun
Résultat :  "failure"
Paramètres : aucun
```

Figure 10 : Syntaxe du nœud "lookup"

Les nœuds de recherche de localisation ont un paramètre obligatoire et deux paramètres facultatifs. Le paramètre obligatoire est "source", la source de la recherche. Ce peut être un URI, ou une valeur non URI. Si la valeur de "source" est un URI, elle indique une localisation que le serveur CPL peut interroger pour obtenir un objet avec le type de support text/uri-list (voir l'enregistrement IANA de ce type, qui apparaît aussi dans la [RFC2483]). L'interrogation est effectuée verbatim, sans ajouter d'informations supplémentaires (comme des paramètres d'URI). Le serveur ajoute les localisations contenues dans cet objet à l'ensemble de localisations.

Les serveurs CPL PEUVENT refuser d'admettre des sources fondées sur un URI pour les interrogations de localisation pour certains schémas d'URI, ou tous. Dans ce cas, ils DEVRAIENT rejeter le script au moment du chargement du script.

On a discuté sur le fait que les serveurs CPL ajoutent des paramètres d'URI à la demande de localisation, afin que (par exemple) les scripts CGI puissent être utilisés pour les résoudre. Cependant, le consensus a été que ce devrait être une extension CPL, et non faire partie de la spécification de base.

Des sources non URL indiquent une source non spécifiée par un URL que le serveur peut interroger pour ajouter des adresses à l'ensemble de localisations. La seule source non URL actuellement définie est "registration", qui spécifie toutes les localisations actuellement enregistrées auprès du serveur.

Le nœud "lookup" a aussi deux paramètres facultatifs. Le paramètre "timeout" spécifie la durée, comme un nombre entier positif de secondes, pendant laquelle le script veut attendre que la recherche soit effectuée. Si il n'est pas spécifié, sa valeur par défaut est 30. Le paramètre "clear" spécifie si l'ensemble de localisations devrait être purgé avant que les nouvelles

localisations soient ajoutées.

La recherche a trois résultats : "success", "notfound", et "failure". Notfound est pris si le processus de recherche a réussi mais n'a donné aucune localisation ; failure est pris si la recherche a échoué pour une raison quelconque, incluant que la durée spécifiée par "timeout" a été dépassée. Si un résultat n'est pas présent, l'exécution du script se termine et le comportement par défaut est effectué.

5.2.1 Usage de "lookup" avec SIP

Pour SIP, la source de recherche "registration" correspond aux localisations enregistrées auprès du serveur en utilisant les messages "REGISTER".

5.3 Suppression de localisation

Un script CPL peut aussi supprimer des localisations de l'ensemble de localisations, en utilisant le nœud "remove-location". La syntaxe de ce nœud est définie à la Figure 11.

La signification de ce nœud dépend du protocole de signalisation sous-jacent.

Nœud : "remove-location"
Résultats : aucun (le prochain nœud suit directement)
Prochain nœud : tout nœud
Paramètres : "location" Localisation à supprimer

Figure 11 : Syntaxe du nœud "remove-location"

Un nœud "remove-location" supprime des localisations de l'ensemble de localisations. Il est surtout utile à la suite d'un nœud "lookup". On en donne un exemple au paragraphe 12.8.

Le nœud "remove-location" a un paramètre facultatif. Le paramètre "location" donne l'URI d'une localisation à retirer de l'ensemble, d'une manière qui dépend du protocole de signalisation. Si ce paramètre n'est pas donné, toutes les localisations sont retirées de l'ensemble.

Le nœud "remove-location" n'a pas d'étiquette de résultat explicite. Dans la syntaxe XML, l'étiquette XML "remove-location" englobe directement l'étiquette du prochain nœud.

5.3.1 Usage de "remove-location" avec SIP

La localisation spécifiée dans le paramètre "location" du nœud "remove-location" est confrontée à l'ensemble de localisations en utilisant les règles standard de confrontation des URI SIP (comme elles sont utilisées, par exemple, pour confronter les adresses de contact lors du rafraîchissement des enregistrements).

6. Opérations de signalisation

Les nœud d'opérations de signalisation causent les événements de signalisation dans le protocole de signalisation sous-jacent. Trois opérations de signalisation sont définies : "proxy" *(mandataire)*, "redirect" *(redirection),* et "reject" *(rejet)*.

6.1 Mandataire

L'opération "proxy" cause la transmission de l'appel déclencheur à l'ensemble de localisations spécifié. La syntaxe du nœud mandataire est données à la Figure 12.

Les événements de signalisation spécifiques invoqués par le nœud "proxy" dépendent du protocole de signalisation, bien que le concept général devrait s'appliquer à tout protocole de signalisation.

Nœud : "proxy"
Résultat : "busy" *(occupé)* Prochain nœud si la tentative d'appel a retourné "busy".
"noanswer" Prochain nœud si la tentative d'appel n'a pas reçu de réponse avant la fin de la temporisation.
"redirection" Prochain nœud si la tentative d'appel a été redirigée.
"failure" Prochain nœud si la tentative d'appel a échoué.

"default" Prochain nœud par défaut pour un résultat non spécifiés.
Paramètres : "timeout" Durée d'essai avant d'abandonner la tentative d'appel.
"recurse" Si il faut faire des redirections de recherches récurrentes.
"ordering" Dans quel ordre essayer l'ensemble de localisations.

Résultat : "busy"
Paramètres : aucun

Résultat : "noanswer" *(pas de réponse)*
Paramètres : aucun

Résultat : "redirection"
Paramètres : aucun

Résultat : "failure" *(échec)*
Paramètres : aucun

Résultat : "default"
Paramètres : aucun

Figure 12 : Syntaxe du nœud "proxy"

Après l'achèvement de l'opération "proxy", le serveur CPL choisit la "meilleure" réponse à la tentative d'appel, comme défini par le protocole de signalisation ou les règles de configuration administrative du serveur.

Si la tentative d'appel a réussi, l'exécution de CPL se termine et le serveur procède à son comportement par défaut (normalement, pour permettre l'établissement de l'appel). Autrement on prend le prochain nœud correspondant à un des résultats du nœud "proxy". Le résultat "busy" est suivi si l'appel était occupé, "noanswer" est suivi si l'appel n'a pas reçu de réponse avant l'expiration du paramètre "timeout" *(fin de temporisation)*, "redirection" est suivi si l'appel a été redirigé, et "failure" est suivi si l'établissement de l'appel a échoué pour une autre raison.

Si une des conditions ci-dessus est vraie, mais si le résultat correspondant n'a pas été spécifié, le résultat "default" *(par défaut)* du nœud "proxy" est plutôt suivi. Si il n'y a pas non plus de nœud "default" spécifié, l'exécution de CPL se termine et le serveur retourne à son comportement par défaut (normalement, pour transmettre la meilleure réponse vers l'amont à l'origine).

Note : les extensions à CPL pour permettre des opérations dans l'appel ou en fin d'appel exigeront un résultat supplémentaire, comme "success" *(succès)*.

Si aucune localisation n'était présente dans l'ensemble, ou si les seules localisations de l'ensemble étaient des localisations auxquelles le serveur ne peut pas mandater un appel (par exemple, les URL "http") on prend le résultat "failure" *(échec)*.

"proxy" a trois paramètres facultatifs. Le paramètre "timeout" spécifie la durée, comme un nombre entier positif de secondes, d'attente que l'appel soit achevé ou rejeté ; après l'écoulement de cette durée, la tentative d'appel est terminée et la branche "noanswer" *(pas de réponse)* est prise. Si ce paramètre n'est pas spécifié, la valeur par défaut est 20 secondes si le nœud "proxy" a spécifié un résultat "noanswer" ou "default" ; autrement le serveur DEVRAIT permettre que l'appel sonne pendant une durée d'une longueur raisonnable (dans la mesure du maximum permis par la politique de ce serveur).

Le second paramètre facultatif est "recurse", qui peut prendre deux valeurs, "oui" ou "non". Cela spécifie si le serveur devrait tenter automatiquement de passer d'autres tentatives d'appel aux adresses de téléphonie dans les réponses de redirection qui ont été retournées du serveur initial. Noter que si la valeur de "recurse" est "oui", le résultat "redirection" n'est jamais pris pour le script. Dans ce cas, ce résultat NE DEVRAIT PAS être présent. La valeur par défaut de ce paramètre est "oui".

Le troisième paramètre facultatif est "ordering". Il peut avoir trois valeurs : "parallel", "sequential", et "first-only". Ce paramètre spécifie dans quel ordre les localisations de l'ensemble de localisations devraient être essayées. "Parallel" demande qu'elles soient toutes essayées simultanément ; "sequential" demande que celle qui a la plus forte priorité soit essayée en premier, celle qui a la plus forte priorité suivante en second, et ainsi de suite, jusqu'à ce qu'une réussisse ou que l'ensemble soit épuisé. "First-only" donne pour instruction au serveur d'essayer seulement l'adresse de plus forte priorité dans l'ensemble, et ensuite de suivre un des résultats. La priorité des localisations dans un ensemble est déterminée par la politique du serveur, bien que les serveurs CPL DEVRAIENT honorer le paramètre "priority" de l'étiquette "location". La valeur par défaut de ce paramètre est "parallel".

Une fois qu'une opération "proxy" est achevée, si le contrôle est passé à d'autres nœuds, toutes les localisations qui ont été utilisées sont purgées de l'ensemble de localisations. C'est-à-dire que l'ensemble de localisations est vidé de ses localisations mandatables si le "ordering" était "parallel" ou "sequential" ; l'élément de plus forte priorité dans l'ensemble est retiré de l'ensemble si "ordering" était "first-only". (Dans tous les cas, les localisations non mandatables comme les URI "http" restent.) Dans le cas d'un résultat "redirection", les nouvelles adresses auxquelles l'appel a été redirigé sont alors ajoutées à l'ensemble de localisations.

6.1.1 Usage de "proxy" avec SIP

Pour SIP, la meilleure réponse à un nœud "proxy" est déterminée par l'algorithme de la spécification SIP. Les résultats du nœud correspondent aux événements suivants :
busy : une réponse 486 ou 600 a été la meilleure réponse reçue pour la demande d'appel.
redirection : une réponse 3xx a été la meilleure réponse reçue pour la demande d'appel.
failure : toute autre réponse 4xx, 5xx, ou 6xx a été la meilleure réponse reçue pour la demande d'appel.
no-answer : aucune réponse finale n'a été reçue pour la demande d'appel avant l'expiration de la temporisation.

Les serveurs SIP DEVRAIENT honorer le paramètre "q" des enregistrements SIP lors de la détermination d'une priorité de localisation.

6.2 Redirect

Redirect fait que le serveur conduit l'appelant à tenter de passer son appel sur l'ensemble de localisations actuellement spécifié. La syntaxe de ce nœud est spécifiée à la Figure 13.

Le comportement spécifique qu'invoque le nœud redirigé dépend du protocole de signalisation sous-jacent impliqué, bien que sa sémantique soit d'application générale.

Nœud : "redirect"
Résultat : aucun (Il peut n'y avoir aucun nœud à suivre.)
Prochain nœud : aucun
Paramètres : "permanent" si la redirection devrait être considérée comme permanente.

Figure 13 : Syntaxe du nœud "redirect"

Redirect termine immédiatement l'exécution du script CPL, de sorte que ce nœud n'a pas de résultats ni de prochain nœud. Il a un paramètre, "permanent", qui spécifie si le résultat retourné devrait indiquer que c'est une redirection permanente. La valeur de ce paramètre est soit "oui" soit "non" et sa valeur par défaut est "non".

6.2.1 Usage de "redirect" avec SIP

Le serveur SIP DEVRAIT envoyer une réponse de classe 3xx à une demande d'appel sur l'exécution d'une étiquette "redirect". Si "permanent" était "oui", le serveur DEVRAIT envoyer la réponse "301" (Déplacement permanent) ; autrement, il DEVRAIT envoyer "302" (Déplacement temporaire).

6.3 Reject

Les nœuds "Reject" causent le rejet par le serveur de toutes les tentatives d'appel. Leur syntaxe est donnée à la Figure 14. Le comportement spécifique qu'ils invoquent dépend du protocole de signalisation sous-jacent impliqué, bien que leur sémantique soit d'application générale.

Nœud : "reject"
Résultat : aucun (Il peut n'y avoir aucun nœud à suivre.)
Prochain nœud : aucun.
Paramètres : "status" code d'état à retourner.
"reason" phrase de cause à retourner.

Figure 14 : Syntaxe du nœud "reject"

Un nœud reject termine immédiatement l'exécution d'un script CPL, donc ce nœud n'a ni résultat ni prochain nœud.

Ce nœud a deux arguments : "status" *(état)* et "reason" *(cause)*. L'argument "status" est exigé, et peut prendre une des

valeurs "busy" *(occupé)*, "notfound" *(pas trouvé)*, "reject" *(rejet)*, "error" *(erreur)*, ou un état défini par le protocole de signalisation.

L'argument "reason" permet facultativement que le script spécifie une cause de rejet.

6.3.1 Usage de "reject" avec SIP

Les serveurs qui mettent en œuvre SIP DEVRAIENT aussi permettre que le champ "status" soit un argument numérique correspondant à un état SIP dans la gamme 4xx, 5xx, ou 6xx.

Ils DEVRAIENT envoyer le paramètre "reason" dans la phrase de cause SIP.

Une transposition suggérée des états désignés est la suivante. Les serveurs PEUVENT utiliser une transposition différente, mais une sémantique similaire DEVRAIT être préservée.

"busy": 486 occupé ici
"notfound": 404 pas trouve
"reject": 603 refus
"error": 500 erreur interne du serveur

7. Opérations qui ne sont pas de signalisation

En plus des opérations de signalisation, CPL définit plusieurs opérations qui n'affectent pas et ne dépendent pas du protocole de signalisation de téléphonie.

7.1 Mail

Le nœud "mail" cause la notification par le serveur à un utilisateur de l'état du script CPL par messagerie électronique. Sa syntaxe est donnée à la Figure 15.

Nœud : "mail"
Résultat : aucun (le prochain nœud suit directement)
Prochain nœud : tout nœud
Paramètres : "url" url Mailto auquel le message devrait être envoyé.

Figure 15 : Syntaxe du nœud "mail"

Le nœud "mail" prend un argument : un URI "mailto" qui donne l'adresse, et tous les paramètres supplémentaires désirés, du message à envoyer. Le serveur envoie le message qui comporte le contenu à l'url concerné ; il DEVRAIT aussi inclure d'autres informations d'état sur la demande d'appel d'origine et le script CPL au moment de la notification.

Utiliser un URL "mailto" complet plutôt que juste une adresse de messagerie électronique permet de spécifier des en-têtes de message supplémentaires comme <mail url="mailto:jones@example.com?subject=Lookup%20failed" />.

Un nœud de messagerie a seulement un résultat possible, car un échec de livraison de message électronique ne peut pas être connu de façon fiable en temps réel. Donc, sa représentation XML n'a pas d'étiquette de résultat : l'étiquette <mail> contient directement une autre étiquette de nœud.

Noter que la syntaxe de XML exige qu'un caractère éperluette , "&", qui est utilisé comme séparateur de paramètre dans les URL "mailto", soit cité comme "&" à l'intérieur des valeurs de paramètres (voir le paragraphe C.12 de la spécification [XML]).

7.1.1 Contenu suggéré pour les informations de messagerie

Ce paragraphe présentes les lignes directrices suggérées pour le message envoyé par suite du nœud "mail", pour les demandes déclenchées par SIP. Le message envoyé par messagerie (déclenché par un protocole quelconque) DEVRAIT contenir toutes ces informations, mais les serveurs PEUVENT choisir d'utiliser un format différent.

1. Si l'URI "mailto" ne spécifie par d'en-tête Subject, le sujet de l'e-mail est "[CPL]", suivi par l'en-tête Subject de la demande SIP. Si l'URI spécifie un en-tête Subject, il est utilisé à la place.

2. Le champ "From" du message électronique est réglé à l'adresse configurée d'un serveur CPL, outrepassant tout champ "From" de l'URI "mailto".

3. Tout en-tête "Reply-To" de l'URI est respecté. Si aucun n'est donné, on utilise alors une version de messagerie électronique du champ d'origine de la demande, si possible (par exemple, un en-tête SIP "From" avec un URI sip: serait converti en adresse e-mail en supprimant le schéma d'URI).

4. Si l'URI "mailto" spécifie un corps, il est utilisé. Si aucun n'était spécifié, le corps DEVRAIT contenir au moins l'identité de l'appelant (à la fois le nom d'affichage et l'adresse de l'appelant) la date et l'heure, l'objet de l'appel et si disponible, la priorité de l'appel.

Le serveur DEVRAIT respecter le langage demandé par l'utilisateur, et envoyer la notification de message en utilisant un langage et un jeu de caractères appropriés.

7.2 Log

Le nœud Log cause l'enregistrement par le serveur des informations sur l'appel dans une mémorisation non volatile. Sa syntaxe est spécifiée à la Figure 16.

Nœud : "log"
Résultats : aucun (le prochain nœud suit directement)
Prochain nœud : tout nœud
Paramètre : "name" Nom du fichier d'enregistrement à utiliser
"comment" Commentaire à placer dans le fichier d'enregistrement

Figure 16 : Syntaxe du nœud "log"

Log prend deux arguments, tous deux facultatifs : "name", qui spécifie le nom de l'enregistrement, et "comment", qui donne un commentaire sur les informations enregistrées. Les serveurs DEVRAIENT aussi inclure d'autres informations dans l'enregistrement, comme l'heure de l'événement enregistré, les informations qui ont déclenché l'appel à enregistrer, et ainsi de suite. Les enregistrements sont spécifiques du propriétaire du script qui enregistre l'événement. Si le paramètre "name" n'est pas donné, l'événement est enregistré sur un fichier standard, défini par le serveur pour le propriétaire du script. La présente spécification ne définit pas comment les utilisateurs peuvent restituer leurs enregistrements du serveur.

Le nom d'un enregistrement n'est qu'un nom logique, et ne correspond pas nécessairement à un fichier physique sur le serveur. L'interprétation du nom du fichier log est défini par le serveur, comme l'est le mécanisme d'accès à ces enregistrements. Le serveur CPL NE DEVRAIT PAS transposer directement les noms d'enregistrements non interprétés en noms de fichiers locaux, pour des raisons de sécurité, de crainte qu'un fichier critique pour la sécurité soit écrasé.

Un serveur CPL fonctionnant correctement NE DEVRAIT jamais permettre que l'événement "log" échoue. À ce titre, les nœuds "log" peuvent avoir seulement un résultat possible, et leur représentation XML n'a pas d'étiquette de résultat explicite. Une étiquette CPL <log> contient directement une autre étiquette de nœud.

8. Sous actions

La syntaxe XML définit une arborescence. Pour permettre des diagramme de flux d'appel plus généraux, et pour permettre la réutilisation et la modularité des scripts, on définit des sous actions. Deux étiquettes sont définies pour les sous actions : définitions de sous action et références de sous action. Leur syntaxe est donnée à la Figure 17.

Étiquette : "subaction"
Sous étiquette : tout nœud
Paramètre : "id" Nom de cette sous action
Pseudo-nœud: "sub"
Résultats : aucun dans l'arborescence XML
Paramètre : "ref" Nom de la sous action à exécuter

Figure 17 : Syntaxe des sous actions et des pseudo-nœuds "sub"

Les sous actions sont définies au moyen des étiquettes "subaction". Ces étiquettes sont placées dans le script CPL après toutes les informations auxiliaires (voir la Section 9) mais avant toute étiquette de niveau supérieur. Elles prennent un

argument "id", jeton qui indique un nom choisi par le script pour la sous action. La valeur "id" pour chaque étiquette "subaction" dans un script DOIT être unique au sein de ce script.

Les sous action sont appelées à partir des étiquettes "sub". L'étiquette "sub" est un "pseudo-nœud", et peut être utilisé en tout endroit d'une action de CPL où un vrai nœud pourrait être utilisé. Elle prend un paramètre, "ref", le nom de la sous action à appeler. L'étiquette "sub" ne contient pas de résultat par elle-même, et son contrôle passe à la sous action.

Les références aux sous actions DOIVENT se rapporter à des sous actions définies avant l'action en cours. Une étiquette "sub" NE DOIT PAS se référer à l'action dans laquelle elle apparaît, ou à une action définie plus tard dans le script CPL. Les actions de niveau supérieur ne peuvent pas être appelées à partir des étiquettes "sub", ou à partir de tout autre moyen. Les serveurs de scripts DOIVENT vérifier au moment de la soumission du script qu'aucun nœud "sub" ne se réfère à une sous action qui n'est pas son prédécesseur adéquat.

Permettre seulement des références arrières aux subs interdit toute forme de récurrence. Une récurrence introduirait la possibilité de scripts CPL sans fin ou non déterministes, une possibilité que nos exigences ont spécifiquement exclue.
Chaque sub DOIT se référer à un identifiant de sous action défini au sein du même script CPL. Aucune liaison externe n'est permise.

Les identifiants de sous action sont sensibles à la casse.

Si une version ou extension ultérieure définit des liens externes, elle devrait probablement utiliser une étiquette différente, peut-être XLink [XLink]. S'assurer de la terminaison en présence de liens externes est un problème difficile.

9. Informations auxiliaires

Aucune information auxiliaire n'est définie dans la spécification CPL de base. Si des informations auxiliaires, qui ne font partie d'aucune opération, se trouvent être nécessaires pour une extension CPL, elles DEVRAIENT être placées dans cette étiquette.

La définition (triviale) de l'étiquette d'informations auxiliaires est donnée à la Figure 18.

Il peut être utile d'inclure les définitions de zone horaire directement dans les scripts CPL, plutôt que de leur faire référence en externe avec les paramètres "tzid" et "tzurl". À cette fin, une extension pourrait être définie pour les inclure ici.

Étiquette : "ancillary"
Paramètres : aucun
Sous étiquette : aucune

Figure 18 : Syntaxe de l'étiquette "ancillary"

10. Comportement par défaut

Lorsque un nœud CPL atteint un résultat non spécifié, soit parce que l'étiquette de résultat n'est pas présente, soit parce que l'étiquette est présente mais ne contient pas de nœud, le comportement du serveur CPL dépend de l'état actuel de l'exécution du script. Cette section donne les opérations qui devraient être effectuées dans chaque cas.

Aucune modification de localisation ou opération de signalisation n'est effectuée, l'ensemble de localisations est vide :
Rechercher la localisation d'usager par tout mécanisme que le serveur utiliserait si aucun script CPL n'était en effet. Mandater, rediriger, ou envoyer un message de rejet, en utilisant toute politique que le serveur utiliserait en l'absence d'un script CPL.

Aucune modification de localisation ou opération de signalisation n'est effectuée, l'ensemble de localisations est non vide :
(Ceci ne peut se produire que sur des appels sortants.) Mandater l'appel aux adresses de l'ensemble de localisations.

Des modifications de localisation sont effectuées, aucune opération de signalisation :
Mandater ou rediriger l'appel, quelle que soit la politique standard du serveur, aux adresses dans l'ensemble actuel de localisations. Si l'ensemble de localisations est vide, retourner un rejet "pas trouvé".

Résultat Pas de réponse du mandataire, pas de fin de temporisation :

(C'est un cas particulier.) Si le résultat "Pas de réponse" d'un nœud mandataire est non spécifié, et qu'aucun paramètre de fin de temporisation n'a été donné au nœud mandataire, l'appel devrait pouvoir sonner pendant la durée maximum permise par le serveur (ou la demande, si la demande spécifiait une temporisation).

Opération de mandataire effectuée précédemment :

Retourner toute "meilleure" réponse de toutes les réponses accumulées pour l'appel à ce point, conformément aux règles du protocole de signalisation sous-jacent.

11. Extensions du CPL

Les serveurs PEUVENT prendre en charge des caractéristiques CPL supplémentaires au delà de celles énumérées dans le présent document. Certaines des extensions qui ont été suggérées sont un moyen d'interroger comment un appel a été authentifié, un contrôle plus riche sur l'adressage H.323, des caractéristiques de système d'extrémité ou spécifiques de l'administrateur, une correspondance d'expression régulière pour les chaînes et adresses, et des contrôle de mi-appel ou de fin d'appel.

Les extensions CPL sont indiquées par l'espace de noms XML [NomsXML]. Chaque extension DOIT avoir un espace de noms XML approprie qui lui est alloué. L'espace de noms XML de l'extension DOIT être différent de l'espace de noms XML défini à la Section 14. L'extension NE DOIT PAS changer la syntaxe ou la sémantique du schéma CPL défini dans le présent document. Toutes les étiquettes et attributs XML qui font partie de l'extension DOIVENT être qualifiés de façon appropriée afin de la placer au sein de cet espace de noms.

Les étiquettes ou attributs dans un script CPL qui sont dans l'espace de noms global (c'est-à-dire, non associés à un espace de noms) sont équivalents aux étiquettes et attributs dans l'espace de noms CPL "urn:ietf:params:xml:ns:cpl".

Un script CPL NE DEVRAIT PAS spécifier d'espace de noms qu'il n'utilise pas. Pour la compatibilité avec les analyseurs qui ne connaissent pas les espaces de noms, un script CPL PEUT omettre l'espace de noms CPL de base pour un script qui n'utilise aucune extension.

Un serveur CPL DOIT rejeter tout script contenant une référence à un espace de noms qu'il ne comprend pas. Il DOIT rejeter tout script contenant une étiquette d'extension ou attribut qui n'est pas qualifié pour être dans un espace de noms approprié.

Une syntaxe telle que

```
<extension-switch>
  <extension has="http://www.example.com/foo">
    [choses étendues]
  </extension>
  <otherwise>
    [choses non étendues]
  </otherwise>
</extension-switch>
```

a été suggérée comme moyen de remplacement du traitement des extensions. Cela permettrait aux scripts d'être chargés sur un serveur sans exiger qu'un auteur de script ait à déterminer quelles extensions sont prises en charge par un serveur. Cependant, l'expérience du développement d'autres langages, notamment Sieve [RFC3028], montre que cela ajoute une complexité excessive aux langages. L'étiquette "extension-switch" pourrait, bien sûr, être elle-même définie dans une extension CPL.

Dans le schéma XML de CPL, on introduit trois éléments abstraits, à savoir 'toplevelaction' *(action de niveau supérieur)*, 'switch' *(commutateur)*, et 'action', qui ont en conséquence le type abstrait 'TopLevelActionType', 'SwitchType', et 'ActionType'. Toute action de niveau supérieur dans une extension CPL DOIT être définie comme le substitutionGroup de l'élément abstrait 'toplevelaction', et avoir le type étendu du 'TopLevelActionType'. Tout commutateur dans une extension CPL DOIT être défini comme substitutionGroup de l'élément abstrait 'switch', et avoir le type étendu de 'SwitchType'. Toute action dans une extension CPL DOIT être définie comme substitutionGroup de l'élément abstrait 'action', et avoir le type étendu de 'ActionType'.

12. Exemples

12.1 Exemple : Call Redirect Unconditional

Le script de la Figure 19 est un simple script qui redirige tous les appels sur une seule localisation fixée.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <incoming>
    <location url="sip:smith@phone.example.com">
      <redirect/>
    </location>
  </incoming>
</cpl>
```

Figure 19 : Exemple de script : Call Redirect Unconditional *(redirection inconditionnelle d'appel)*

12.2 Exemple : Call Forward Busy/No Answer

Le script de la Figure 20 illustre un comportement plus complexe. On voit qu'un mandataire initial tente une adresse, avec plus d'opérations si cela échoue. On voit aussi comment plusieurs résultats prennent la même arborescence d'action, par l'utilisation de sous actions.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <subaction id="voicemail">
    <location url="sip:jones@voicemail.example.com">
      <proxy/>
    </location>
  </subaction>
  <incoming>
    <location url="sip:jones@jonespc.example.com">
      <proxy timeout="8">
        <busy>
          <sub ref="voicemail"/>
        </busy>
        <noanswer>
          <sub ref="voicemail"/>
        </noanswer>
      </proxy>
    </location>
  </incoming>
</cpl>
```

Figure 20 : Exemple de script : Call Forward Busy/No Answer *(transmisssion d'appel sur occupation/non réponse)*

12.3 Exemple : Call Forward: Redirect et Default

Le script de la Figure 21 illustre plus de comportement de mandataire. Le serveur essaye initialement de mandater à une seule adresse. Si cette tentative est redirigée, une nouvelle redirection est générée en utilisant la localisation retournée. Dans tous les autres cas d'échec pour le nœud mandataire, une opération par défaut – transmission à une boîte vocale – est effectuée.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <incoming>
```

```
   <location url="sip:jones@jonespc.example.com">
    <proxy>
     <redirection>
      <redirect/>
     </redirection>
     <default>
      <location url="sip:jones@voicemail.example.com">
       <proxy/>
      </location>
     </default>
    </proxy>
   </location>
  </incoming>
</cpl>
```

Figure 21 : Exemple de script : Call Forward : Redirect et Default

12.4 Exemple : Call Screening

Le script de la Figure 22 illustre les commutateurs d'adresses et le rejet d'appel, sous la forme d'un script de filtrage d'appel. Noter aussi que parce que il manque une clause "otherwise" au commutateur d'adresses, si le schéma initial ne correspond pas, le script ne définit aucune opération. Le serveur procède donc à son comportement par défaut, qui va probablement être de contacter l'usager.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <incoming>
    <address-switch field="origin" subfield="user">
      <address is="anonymous">
        <reject status="reject" reason="Je rejette les appels anonymes"/>      </address>
    </address-switch>
  </incoming>
</cpl>
```

Figure 22 : Exemple de script : Call Screening *(filtrage d'appel)*

12.5 Exemple : acheminement sur priorité et langage

Le script de la Figure 23 illustre un choix de service sur la base de la valeur de priorité de l'appel et des réglages de langue. Si la demande d'appel a une priorité de "urgent" ou plus fort, le comportement de script par défaut est effectué. Autrement, le champ Language est examiné à la recherche de l'étiquette de langue "es" (espagnol). Si elle est présente, l'appel est mandaté à un opérateur hispanophone ; les autres appels sont mandatés à un opérateur anglophone.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <incoming>
    <priority-switch>
      <priority greater="urgent"/>
      <otherwise>
        <language-switch>
          <language matches="es">
            <location url="sip:spanish@operator.example.com">
              <proxy/>
            </location>
          </language>
          <otherwise>
            <location url="sip:english@operator.example.com">
              <proxy/>
```

```
          </location>
         </otherwise>
        </language-switch>
      </otherwise>
    </priority-switch>
  </incoming>
</cpl>
```

Figure 23 : Exemple de script : acheminement sur priorité et langage

12.6 Exemple : filtrage d'appel sortant

Le script de la Figure 24 illustre un script qui filtre les appels sortants, sous la forme d'un script qui empêche les appels au 1-900 (service premium). Ce script illustre aussi la correspondance de sous domaines.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <outgoing>
    <address-switch field="original-destination" subfield="tel">
      <address subdomain-of="1900">
        <reject status="reject"
          reason="Les appels 1-900 ne sont pas permis."/>
      </address>
    </address-switch>
  </outgoing>
</cpl>
```

Figure 24 : Exemple de script : filtrage d'appel sortant

12.7 Exemple : acheminement selon l'heure du jour

La Figure 25 illustre des conditions fondées sur l'heure et les zones horaires.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <incoming>
    <time-switch tzid="America/New_York"
      tzurl="http://zones.example.com/tz/America/New_York">
      <time dtstart="20000703T090000" duration="PT8H" freq="weekly"
        byday="MO,TU,WE,TH,FR">
        <lookup source="registration">
          <success>
            <proxy/>
          </success>
        </lookup>
      </time>
      <otherwise>
        <location url="sip:jones@voicemail.example.com">
          <proxy/>
        </location>
      </otherwise>
    </time-switch>
  </incoming>
</cpl>
```

Figure 25 : Exemple de script : acheminement selon l'heure du jour

12.8 Exemple : filtrage de localisation

La Figure 26 illustre les opérations de filtrage sur l'ensemble de localisations. Dans cet exemple, on suppose que la version 0.9beta2 de "agent d'utilisateur SIP de logiciel inadéquat" a mal mis en œuvre certains dispositifs, et qu'on doit donc résoudre ces problèmes. Nous savons qu'il ne peut pas réussir à parler avec un certain appareil mobile qu'on peut avoir enregistré, de sorte qu'on retire cette localisation de l'ensemble de localisations. Une fois cette opération terminée, l'établissement d'appel peut se poursuivre normalement.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <incoming>
    <string-switch field="user-agent">
      <string is="Inadequate Software SIP User Agent/0.9beta2">
        <lookup source="registration">
          <success>
            <remove-location location="sip:me@mobile.provider.net">
              <proxy/>
            </remove-location>
          </success>
        </lookup>
      </string>
    </string-switch>
  </incoming>
</cpl>
```

Figure 26 : Exemple de script : filtrage de localisation

12.9 Exemple : opérations qui ne sont pas de signalisation

La Figure 27 illustre des opérations de non signalisation; en particulier, d'alerte d'un usager par messagerie électronique si le serveur de recherche a échoué. La principale motivation pour avoir le nœud "mail" est de permettre cette sorte de notification hors bande des conditions d'erreur, car l'usager pourrait autrement ignorer tous les problèmes.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <incoming>
    <lookup
       source="http://www.example.com/cgi-bin/locate.cgi?user=mary"
       timeout="8">
      <success>
        <proxy/>
      </success>
      <failure>
        <mail url="mailto:mary@example.com?subject=Lookup%20failed"/>
      </failure>
    </lookup>
  </incoming>
</cpl>
```

Figure 27 : Exemple de script : opérations qui ne sont pas de signalisation

12.10 Exemple : extensions hypothétiques

L'exemple de la Figure 28 montre une extension hypothétique qui met en œuvre des sonneries distinctives. L'espace de noms XML "http://www.example.com/distinctive-ring" spécifie un nouveau nœud nommé "ring".

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.example.com/distinctive-ring"
  xmlns="http://www.example.com/distinctive-ring"
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:CPL="urn:ietf:params:xml:ns:cpl"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:import namespace="urn:ietf:params:xml:ns:cpl"
     schemaLocation="cpl.xsd"/>
  <xs:complexType name="DRingAction">
   <xs:complexContent>
    <xs:extension base="CPL:ActionType">
     <xs:attribute name="ringstyle" type="xs:string"
        use="optional"/>
    </xs:extension>
   </xs:complexContent>
  </xs:complexType>
  <xs:element name="ring" type="DRingAction"
     substitutionGroup="CPL:action"/>
</xs:schema>

<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:dr="http://www.example.com/distinctive-ring"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd
     http://www.example.com/distinctive-ring distinctive-ring.xsd">
  <incoming>
   <address-switch field="origin">
    <address is="sip:boss@example.com">
     <dr:ring ringstyle="warble"/>
    </address>
   </address-switch>
  </incoming>
</cpl>
```

Figure 28 : Exemple de schéma et de script : extension hypothétique de sonnerie distinctive

L'exemple de la Figure 29 met en œuvre un nouvel attribut hypothétique pour les commutateurs d'adresses, pour permettre des correspondances d'expressions régulières. Il définit un nouvel attribut "regex" pour le nœud standard "address".

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <incoming>
   <address-switch field="origin" subfield="user"
      xmlns:re="http://www.example.com/regex">
    <address re:regex="(.*.smith|.*.jones)">
     <reject status="reject"
        reason="je ne veux pas parler aux Smith ou aux Jones"/>
    </address>
   </address-switch>
  </incoming>
</cpl>
```

Figure 29 : Exemple de script : extension hypothétique d'expression régulière

12.11 Exemple : exemple complexe

Finalement, la Figure 30 donne un exemple complexe du comportement sophistiqué qui peut être réalisé en combinant des nœuds CPL. Dans ce cas, l'usager tente de faire que ses appels atteignent son bureau ; si il ne répond pas dans un bref délai, les appels de son patron sont transmis à son mobile, et tous les autres appels sont dirigés sur la messagerie vocale. Si l'établissement d'appel échoue, aucune opération n'est spécifiée, de sorte que le comportement par défaut du serveur est

effectué.

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <subaction id="voicemail">
    <location url="sip:jones@voicemail.example.com">
      <redirect />
    </location>
  </subaction>
  <incoming>
    <location url="sip:jones@phone.example.com">
      <proxy timeout="8">
        <busy>
          <sub ref="voicemail" />
        </busy>
        <noanswer>
          <address-switch field="origin">
            <address is="sip:boss@example.com">
              <location url="tel:+19175551212">
                <proxy />
              </location>
            </address>
            <otherwise>
              <sub ref="voicemail" />
            </otherwise>
          </address-switch>
        </noanswer>
      </proxy>
    </location>
  </incoming>
</cpl>
```

Figure 30 : Exemple de script complexe

13. Considérations pour la sécurité

CPL est conçu de façon à permettre de spécifier les services d'une manière qui empêche des scripts potentiellement hostiles ou mal configurés de lancer des attaques contre la sécurité, y compris des attaques de déni des service. Parce que le temps d'exécution du script est strictement acyclique, et parce que le nombre d'opérations de script possibles est strictement limité, les scripts ne devraient pas être capables d'infliger des dommages à un serveur CPL.

Parce que les scripts peuvent diriger les appels téléphoniques des utilisateurs, la méthode par laquelle les scripts sont transmis d'un client à un serveur DOIT avoir une authentification forte. Une telle méthode n'est pas spécifiée dans le présent document.

Les serveurs de scripts DEVRAIENT permettre aux administrateurs de serveurs de contrôler les détails de quelles opérations CPL sont permises.

14. Considérations relatives à l'IANA

Le présent document enregistre un nouveau type MIME, application/cpl+xml, et un nouvel URN selon la [RFC2141], la [RFC2648], et la [RFC3688].

L'espace de noms XML urn:ietf:params:xml:ns:cpl se réfèrera seulement à la version de CPL du présent document et ne changera pas. Toutes les améliorations à CPL DOIVENT être faites par des extensions et DOIVENT avoir des espaces de noms différents.

14.1 Enregistrement du sous espace d'URN pour urn:ietf:params:xml:ns:cpl

URI: urn:ietf:params:xml:ns:cpl

Contact d'enregistrement :
Jonathan Lennox <lennox@cs.columbia.edu>
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML :

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
   "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
    content="text/html;charset=iso-8859-1"/>
  <title>Call Processing Language Namespace</title>
</head>
 <body>
  <h1>Namespace for Call Processing Language</h1>
  <h2>urn:ietf:params:xml:ns:cpl</h2>
  <p><a href="ftp://ftp.rfc-editor.org/in-notes/rfc3880.txt">
      RFC3880</a>.</p>
</body>
</html>
END
```

14.2 Enregistrement de schéma

La présente spécification enregistre le schéma XML pour CPL, selon les lignes directrices de la [RFC3688].

URI : urn:ietf:params:xml:schema:cpl

Contact d'enregistrement :
Jonathan Lennox <lennox@cs.columbia.edu>
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

XML: L'XML se trouve à l'Appendice C.

14.3 Enregistrement MIME

En tant que type XML, l'enregistrement MIME de CPL se conforme aux "types de supports XML" de la [RFC3023].

Nom de type de support MIME : application
Nom de sous type MIME : cpl+xml
Paramètres obligatoires : aucun
Paramètres facultatifs : charset. Comme pour application/xml dans la RFC 3023.
Considérations de codage : comme pour application/xml dans la RFC 3023.
Considérations de sécurité : voir la Section 13, et la Section 10 de la RFC 3023.
Considérations d'interopérabilité : des serveurs CPL différents peuvent utiliser des types d'adresse incompatibles. Cependant, toutes les questions potentielles d'interopérabilité devraient être résolues au moment du chargement d'un script ; il ne devrait pas y avoir de problème d'interopérabilité détecté au moment du démarrage.
Spécification publiée : le présent document.
Applications qui utilisent ce type de support : les serveurs mandataires SIP et les autres serveurs de téléphonie, et les logiciels clients pour contrôler leur comportement.
Informations supplémentaires :
Numéro magique : aucun
Extension de fichier : .cpl ou .xml
Code de type de fichier Macintosh : "TEXT"

Adresse de messagerie des personnes à contacter pour plus d'informations :
Jonathan Lennox <lennox@cs.columbia.edu>
Xiaotao Wu <xiaotaow@cs.columbia.edu>
Henning Schulzrinne <hgs@cs.columbia.edu>

Utilisation prévue : Courante
Auteur/Contrôleur des changements : l'IETF.

15. Remerciements

Le présent document a été revu et commenté par le groupe de travail Téléphonie IP de l'IETF. Nous remercions spécifiquement les personnes suivantes de leur aide : le script d'examen d'appel sortant a été écrit par Kenny Hom ; Paul E. Jones a grandement contribué à la transposition des adresses H.323 ; le texte de la section time-switch a été tiré (légèrement modifié) de la [RFC2445] de Frank Dawson et Derik Stenerson ; on s'est beaucoup inspiré, notamment pour le manque de complétude au sens de Turing du langage et pour la syntaxe de correspondance de chaîne, de la spécification de Sieve [RFC3028], un langage pour le filtrage d'utilisateur de la messagerie électronique. Thomas F. La Porta et Jonathan Rosenberg ont fourni de très utiles discussions, contributions, et suggestions. Richard Gumpertz a effectué une révision technique et rédactionnelle de dernière minute très utile de la spécification.

A. Algorithme de résolution des commutateurs horaires

L'algorithme qui suit détermine si un certain instant tombe dans une répétition d'une récurrence de "commutateur horaire". Si le pré traitement décrit au paragraphe 4.4.1 a été fait, il opère en temps constant. Le code libre Java qui met en œuvre cet algorithme est disponible à http://www.cs.columbia.edu/~lennox/Cal-Code/ sur la Toile mondiale.

Cet algorithme est estimé correct, mais la présente section est non normative. Le paragraphe 4.4, et la [RFC2445], sont les définitions définitives des récurrences.

1. Calculer l'heure de l'appel, dans la zone horaire du commutateur horaire.
2. Si l'heure de l'appel est plus tôt que "dtstart", échec NOMATCH.
3. Si l'heure de l'appel est moins que "duration" après dtstart, réussite MATCH.
4. Déterminer la plus petite unité spécifiée dans une règle "byxxx" ou par le "freq". On appelle cela Unité minimum. Déterminer l'instant précédent (avant, ou égal à, l'heure de l'appel) lorsque toutes les unités de temps plus petites que l'unité minimum sont les mêmes que celles de "dtstart". Si l'unité minimum est une seconde, cette heure est la même que l'instant. Si l'unité minimum est une minute ou une heure, respectivement les minutes ou les minutes et heures, doivent être les mêmes que "dtstart". Pour toutes les autres unités minimum, l'heure doit être la même que "dtstart". Si l'unité minimum est une semaine, le jour de la semaine doit être le même que "dtstart". Si l'unité minimum est un mois, le jour du mois doit être le même que "dtstart". Si l'unité minimum est un an, le mois et le jour du mois doivent tous deux être le même que "dtstart". (Noter que cela signifie qu'il peut être nécessaire de revenir en arrière de plus d'une unité minimum – si l'unité minimum est un mois, certains mois n'ont pas de 31ème (ou 30ème ou 29ème) jour ; si l'unité minimum est un an, certaines années n'ont pas de 29 février. Dans le calendrier grégorien, il n'est jamais nécessaire de revenir en arrière de plus de deux mois si l'unité minimum est un mois, ou de huit ans si l'unité minimum est un an. Entre 1904 et 2096, il n'est jamais nécessaire de revenir en arrière de plus de quatre ans – le retour en arrière de huit ans ne peut survenir que lorsque le calendrier grégorien "saute" une année bissextile. Appelons cet instant le candidat heure de début.

5. Si le temps entre le candidat heure de début et l'heure de l'appel est plus que la durée, échec NOMATCH.

6. Si le candidat heure de début est plus tard que le paramètre "until" de la récurrence (ou le "until" virtuel calculé hors ligne à partir de "count"), échec NOMATCH.

7. Appelons l'unité du paramètre "freq" de la récurrence Unité de fréquence. Déterminer l'unité de fréquence qui englobe le candidat heure de début, et celle qui englobe "dtstart". Calculer le nombre d'unités de fréquence qui sont passées entre ces deux instants. Si ce n'est pas un multiple du paramètre "interval", échec NOMATCH.

8. Pour chaque règle "byxxx", confirmer que le candidat heure de début correspond à une des options spécifiées par cette règle "byxxx". S'il en est ainsi, réussite MATCH.

9. Calculer un candidat heure de début précédent. Répéter jusqu'à ce que la différence entre le candidat heure de début et l'heure de l'appel soit plus que la durée. Si aucun candidat heure de début n'a été validé, échec NOMATCH.

B. Usage suggéré de CPL avec H.323

Cet appendice suggère une utilisation de CPL avec H.323 [H.323]. Le groupe d'études 16 de l'UIT, qui a développé H.323, propose de travailler sur une transposition officielle de CPL pour ce protocole. Cette section n'est donc pas normative.

B.1 Usage de "address-switch" avec H.323

Les commutateurs d'adresses sont spécifiés au paragraphe 4.1. Cette section spécifie la transposition entre les messages H.323 et les champs et sous champs de "address-switch".

Pour H.323, l'adresse "origin" correspond aux adresses d'alias dans le champ "sourceAddress" de l'élément d'information "Setup-UUIE" d'usager à usager, et à l'élément d'information [Q.931] "Numéro d'appelant". Si les deux champs sont présents, ou si plusieurs adresses d'alias sont présentes pour "sourceAddress", celle qui a la priorité est l'affaire de la politique locale ; le serveur DEVRAIT utiliser la même résolution qu'il utiliserait pour les décisions d'acheminement dans ce cas. De même, l'adresse de "destination" correspond aux adresses d'alias du champ "destinationAddress", et à l'élément d'information Q.931 "Numéro du demandé".

L'adresse "original-destination" correspond à l'élément d'information Q.931 "Numéro de redirection", si il est présent; autrement, il est le même que l'adresse "destination".

La transposition des adresses H.323 en sous champs dépend du type de l'adresse d'alias. Un type de sous champ supplémentaire, "alias-type", est défini pour les serveurs H.323, qui correspond au type de l'adresse. Les valeurs possibles sont "dialedDigits", "h323-ID", "url-ID", "transportID", "email-ID", "partyNumber", "mobileUIM", et "Q.931IE". Si des versions futures de la spécification H.323 définissent des types supplémentaires d'adresses d'alias, ces noms PEUVENT aussi être utilisés.

Dans les versions de H.323 antérieures à la version 4, "dialedDigits" était appelé "e164". Les deux noms DEVRAIENT être traités comme synonymes.

La valeur du sous champ "address-type" pour les messages H.323 est "h323" sauf si le type d'alias est "url-ID" et le schéma d'URL est quelque chose d'autre que h323 ; dans ce cas, le type d'adresse est le schéma d'URL, comme spécifié au paragraphe 4.1.1 pour SIP.

Un serveur CPL à capacité H.323 DEVRAIT transposer les sous champs d'adresse à partir de l'alias principal utilisé pour l'acheminement. Il PEUT aussi transposer les sous champs à partir d'autres alias, si les sous champs dans l'adresse principale ne sont pas présents.

Les transpositions suivantes sont utilisées pour les types d'alias H.323 :

dialedDigits, partyNumber, mobileUIM, et Q.931IE : les sous champs "tel" et "user" sont la chaîne de chiffres, comme l'est la forme "entire-address". Les sous champs "host" et "port" ne sont pas présents.

url-ID : on utilise les mêmes transpositions que pour SIP, au paragraphe 4.1.1.

h323-ID : le champ "user" est la chaîne de caractères, comme l'est la forme "entire-address". Tous les autres sous champs ne sont pas présents.

email-ID : les sous champs "user" et "host" sont réglés aux parties correspondantes de l'adresse de messagerie électronique. Les sous champs "port" et "tel" ne sont pas présents. La forme "entire-address" correspond à l'adresse de messagerie électronique entière.

transportID : si TransportAddress est du type "ipAddress," "ipSourceRoute," ou "ip6Address", le sous champ "host" est réglé à l'élément "ip" de la séquence, traduite en représentation textuelle standard IPv4 ou IPv6, et le sous champ "port" est réglé à l'élément "port" de la séquence représentée en décimal. Les champs "tel" et "user" ne sont pas présents. La forme "entire-address" n'est pas définie. La représentation et la transposition des adresses de transport n'est pas définie pour les adresses non IP.

[H.323] définit un schéma d'URI "h323". Cet appendice définit une transposition pour ces URI sur le sous champ CPL "address-switch", comme donné au paragraphe 4.1. Cette définition est aussi disponible comme [RFC3508], qui est un extrait de la spécification H.323.

Pour les URI h323, les sous champs "user", "host", et "port" sont réglés à la partie correspondante de l'URI H.323. Le sous champ "tel" n'est pas présent. La forme "entire-address" correspond à l'URI entier.

Cette transposition PEUT être utilisée pour les URI h323 dans les alias d'adresse h323 "url-ID", et pour les URI h323 dans les messages SIP.

B.2 Usage de "string-switch" avec H.323

Pour H.323, le nœud "string-switch" (voir le paragraphe 4.2) est utilisé comme suit. Le champ "display" correspond à l'élément d'information Q.931 du même nom, copié verbatim. Les champs "subject", "organization", et "user-agent" ne sont pas utilisés et ne sont jamais présents.

L'élément d'information "display" est utilisé conventionnellement pour les besoins de l'identifiant d'appelant, donc il devrait être transposé dans le sous champ "display" d'un "address-match" avec le champ "originator". Cependant, comme a) il est un élément d'information de niveau message, et non de niveau adresse, et b) la spécification [Q.931] dit seulement que "l'objet de l'élément d'information Display est de fournir des informations d'affichage à l'utilisateur", il semble être plus approprié de lui permettre plutôt d'être confronté dans un "string-switch".

B.3 Usage de "language-switch" avec H.323

Les gammes de langage pour le commutateur "language-switch" sont obtenues de l'UUIE H.323 "language". Le commutateur n'est pas présent si le message initial ne contenait pas cet UUIE.

B.4 Usage de "priority-switch" avec H.323

Tous les messages H.323 sont considérés comme ayant la priorité "normal" pour les besoins d'un commutateur de priorité (voir le paragraphe 4.5).

B.5 Usage de "location" avec H.323

Les localisations dans les nœuds de localisation explicites (paragraphe 5.1) sont spécifiés comme des URL. Donc, toutes les localisations ajoutées de cette manière sont interprétées comme étant du type d'alias "url-ID" dans H.323.

Les spécifications d'autres types d'alias d'adresse H.323 exigeront une extension CPL (voir la Section 11).

B.6 Usage de "lookup" avec H.323

Pour les nœuds de recherche de localisation (paragraphe 5.2) la source de recherche "registration" correspond aux localisations enregistrées auprès du serveur en utilisant les messages "RAS".

B.7 Usage de "remove-location" avec H.323

Les nœuds de suppression de localisation (paragraphe 5.3) suppriment les adresses qui ont le type d'alias "url-ID" en utilisant la confrontation de chaîne verbatim sur les URL. Si un URL "tel" est spécifié comme localisation, les adresses correspondantes (en ignorant les séparateurs visuels) avec les types d'alias "dialedDigits" ("e164"), "partyNumber", "mobileUIM", ou "Q.931IE" sont aussi supprimées. Aucun mécanisme n'est fourni pour supprimer d'autres types d'alias.

C. Schéma XML pour CPL

Cette section comporte un schéma XML complet qui décrit la syntaxe XML de CPL. Chaque script soumis à un serveur CPL DEVRAIT se conformer à ce schéma XML. Lors de l'analyse de la conformité des scripts avec le DTD CPL dans les documents antérieurs, les lignes DOCTYPE dans les scripts devraient être ignorées. Noter que la conformité à ce schéma n'est pas une condition suffisante pour qu'un script CPL soit correct, car beaucoup des conditions décrites dans cette spécification ne sont pas exprimables dans une syntaxe de schéma. La Figure 31 montre la structure du schéma. 'incoming' *(entrant)* et 'outgoing' *(sortant)* sont définis comme le substitutionGroup *(groupe de substitution)* de 'toplevelaction' *(action de niveau supérieur)*. Tous les commutateurs sont définis comme substitutionGroup de l'élément 'switch'. Toutes les actions sont définies comme substitutionGroup de l'élément 'action'.

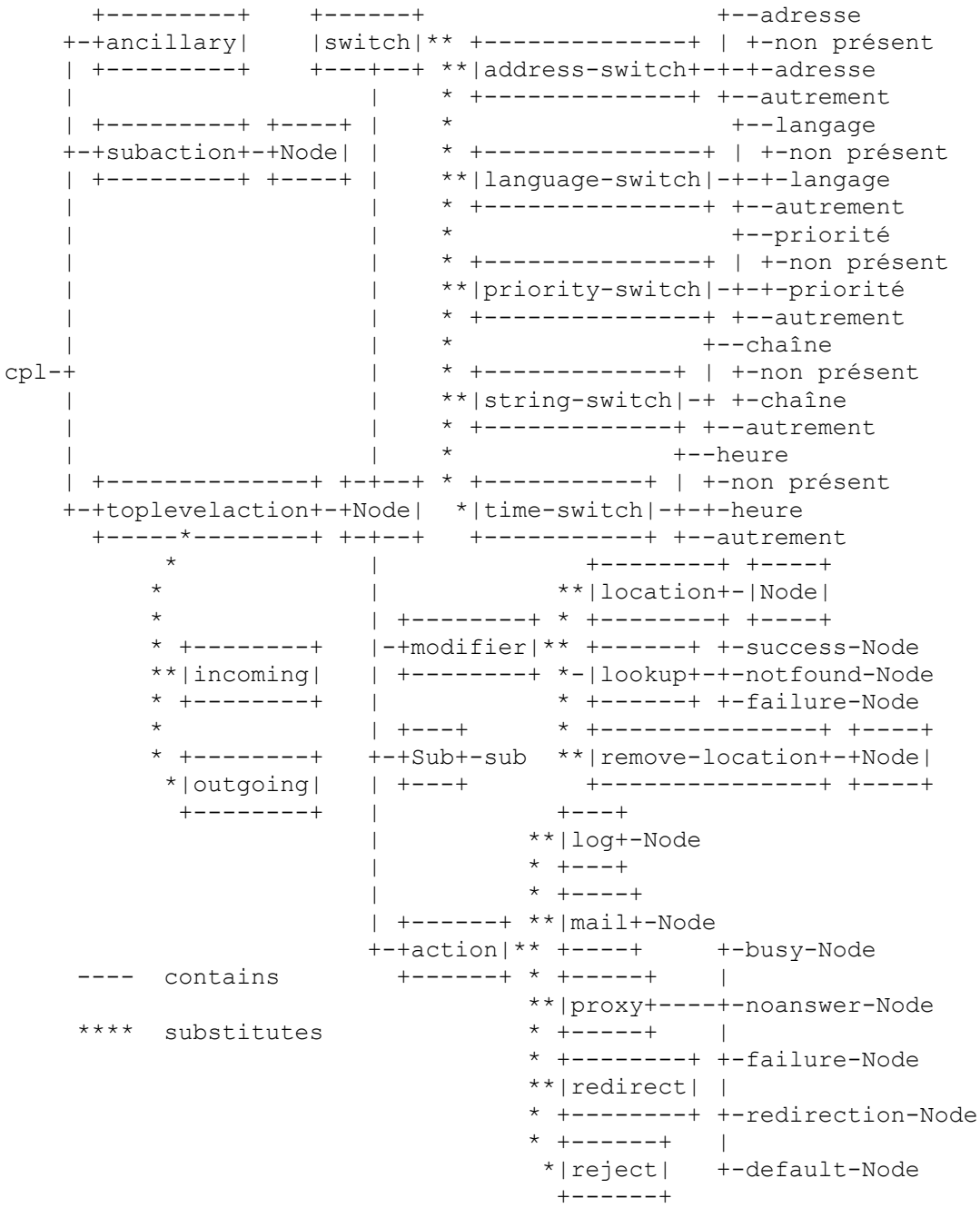

Figure 31 : Structure du schéma XML pour CPL

```
BEGIN
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:cpl"
 xmlns="urn:ietf:params:xml:ns:cpl"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">
 <xs:complexType name="TopLevelActionType" abstract="true">
  <xs:group ref="Node"/>
 </xs:complexType>
 <xs:element name="toplevelaction" type="TopLevelActionType"/>
 <xs:complexType name="ActionType" abstract="true"/>
 <xs:element name="action" type="ActionType"/>
 <xs:complexType name="SwitchType" abstract="true"/>
 <xs:element name="switch" type="SwitchType"/>
 <xs:complexType name="ModifierType" abstract="true"/>
```

```
<xs:element name="modifier" type="ModifierType"/>
<xs:element name="location" type="LocationType"
   substitutionGroup="modifier"/>
<xs:element name="lookup" type="LookupType"
   substitutionGroup="modifier"/>
<xs:element name="remove-location" type="RemoveLocationType"
   substitutionGroup="modifier"/>
<xs:element name="sub" type="SubAction"/>
<xs:group name="Node">
 <xs:choice>
  <xs:element ref="switch" minOccurs="0" maxOccurs="1"/>
  <xs:element ref="modifier" minOccurs="0" maxOccurs="1"/>
  <xs:element ref="sub" minOccurs="0" maxOccurs="1"/>
  <xs:element ref="action" minOccurs="0" maxOccurs="1"/>
 </xs:choice>
</xs:group>
<xs:complexType name="OtherwiseAction">
 <xs:group ref="Node"/>
</xs:complexType>
<xs:complexType name="NotPresentAction">
 <xs:group ref="Node"/>
</xs:complexType>
<xs:simpleType name="YesNoType">
 <xs:restriction base="xs:NMTOKEN">
  <xs:enumeration value="yes"/>
  <xs:enumeration value="no"/>
 </xs:restriction>  </xs:simpleType>
<xs:simpleType name="StatusType">
 <xs:union>
  <xs:simpleType>
   <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="busy"/>
    <xs:enumeration value="notfound"/>
    <xs:enumeration value="reject"/>
    <xs:enumeration value="error"/>
   </xs:restriction>
  </xs:simpleType>
  <xs:simpleType>
   <xs:restriction base="xs:string"/>
  </xs:simpleType>
 </xs:union>
</xs:simpleType>
<xs:simpleType name="OrderingType">
 <xs:restriction base="xs:NMTOKEN">
  <xs:enumeration value="parallel"/>
  <xs:enumeration value="sequential"/>
  <xs:enumeration value="first-only"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="AddressFieldType">
 <xs:union>
  <xs:simpleType>
   <xs:restriction base="xs:NMTOKEN">
    <xs:enumeration value="origin"/>
    <xs:enumeration value="destination"/>
    <xs:enumeration value="original-destination"/>
   </xs:restriction>
  </xs:simpleType>
  <xs:simpleType>
   <xs:restriction base="xs:string"/>
  </xs:simpleType>
 </xs:union>
</xs:simpleType>
```

```
 <xs:simpleType name="AddressSubfieldType">
  <xs:union>
   <xs:simpleType>
    <xs:restriction base="xs:NMTOKEN">
     <xs:enumeration value="address-type"/>
     <xs:enumeration value="user"/>
     <xs:enumeration value="host"/>
     <xs:enumeration value="port"/>
     <xs:enumeration value="tel"/>
     <xs:enumeration value="display"/>
     <xs:enumeration value="password"/>
     <xs:enumeration value="alias-type"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:simpleType>
    <xs:restriction base="xs:string"/>
   </xs:simpleType>
  </xs:union>
 </xs:simpleType>
 <xs:complexType name="AddressType">
  <xs:annotation>
<xs:documentation>Exactement un de ces trois attributs doit apparaître</xs:documentation>
  </xs:annotation>    <xs:group ref="Node"/>
  <xs:attribute name="is" type="xs:string" use="optional"/>
  <xs:attribute name="contains" type="xs:string" use="optional">
   <xs:annotation>
<xs:documentation>pour "display" seulement</xs:documentation>
   </xs:annotation>
  </xs:attribute>
  <xs:attribute name="subdomain-of" type="xs:string"
    use="optional">
   <xs:annotation>
    <xs:documentation>for "host", "tel" only</xs:documentation>
   </xs:annotation>
  </xs:attribute>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
 </xs:complexType>
 <xs:complexType name="AddressSwitchType">
  <xs:complexContent>
   <xs:extension base="SwitchType">
    <xs:sequence>
     <xs:element name="address" type="AddressType" minOccurs="0"
       maxOccurs="unbounded"/>
     <xs:sequence minOccurs="0">
      <xs:element name="not-present" type="NotPresentAction"/>
      <xs:element name="address" type="AddressType"
        minOccurs="0" maxOccurs="unbounded"/>
     </xs:sequence>
     <xs:element name="otherwise" type="OtherwiseAction"
       minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="field" type="AddressFieldType"
      use="required"/>
    <xs:attribute name="subfield" type="AddressSubfieldType"
      use="optional"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:element name="address-switch" type="AddressSwitchType"
   substitutionGroup="switch"/>
 <xs:simpleType name="StringFieldType">
  <xs:restriction base="xs:NMTOKEN">
   <xs:enumeration value="subject"/>
```

```
      <xs:enumeration value="organization"/>
      <xs:enumeration value="user-agent"/>
      <xs:enumeration value="display"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="StringType">
    <xs:group ref="Node"/>
    <xs:attribute name="is" type="xs:string" use="optional"/>
    <xs:attribute name="contains" type="xs:string" use="optional"/>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="StringSwitchType">
    <xs:complexContent>
      <xs:extension base="SwitchType">
        <xs:sequence>
          <xs:element name="string" type="StringType" minOccurs="0"
            maxOccurs="unbounded"/>
          <xs:sequence minOccurs="0">
            <xs:element name="not-present" type="NotPresentAction"/>
            <xs:element name="string" type="StringType" minOccurs="0"
              maxOccurs="unbounded"/>
          </xs:sequence>
        <xs:element name="otherwise" type="OtherwiseAction"
            minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="field" type="StringFieldType"
          use="required">
          <xs:annotation>
<xs:documentation>Les chaînes sont confrontées comme chaînes Unicode insensibles à la casse.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="string-switch" type="StringSwitchType"
    substitutionGroup="switch"/>
  <xs:complexType name="LanguageType">
    <xs:group ref="Node"/>
    <xs:attribute name="matches" type="xs:string" use="required">
      <xs:annotation>
<xs:documentation>La valeur d'un de ces paramètres est une étiquette de langage, comme défini dans la RFC3066.
</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="LanguageSwitchType">
    <xs:complexContent>
      <xs:extension base="SwitchType">
        <xs:sequence>
          <xs:element name="language" type="LanguageType"
            minOccurs="0" maxOccurs="unbounded"/>
          <xs:sequence minOccurs="0">
            <xs:element name="not-present" type="NotPresentAction"/>
            <xs:element name="language" type="LanguageType"
              minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:element name="otherwise" type="OtherwiseAction"
            minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
 <xs:element name="language-switch" type="LanguageSwitchType"
    substitutionGroup="switch"/>
 <xs:simpleType name="FreqType">
  <xs:restriction base="xs:NMTOKEN">
   <xs:pattern value="[s|S][e|E][c|C][o|O][n|N][d|D][l|L][y|Y]"/>
   <xs:pattern value="[m|M][i|I][n|N][u|U][t|T][e|E][l|L][y|Y]"/>
   <xs:pattern value="[h|H][o|O][u|U][r|R][l|L][y|Y]"/>
   <xs:pattern value="[d|D][a|A][i|I][l|L][y|Y]"/>
   <xs:pattern value="[w|W][e|E][e|E][k|K][l|L][y|Y]"/>
   <xs:pattern value="[m|M][o|N][n|N][t|T][h|H][l|L][y|Y]"/>
   <xs:pattern value="[y|Y][e|E][a|A][r|R][l|L][y|Y]"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="YearDayType">
  <xs:union>
   <xs:simpleType>
    <xs:restriction base="xs:integer">
     <xs:minInclusive value="-366"/>
     <xs:maxInclusive value="-1"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:simpleType>
   <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxExclusive value="366"/>
    </xs:restriction>
   </xs:simpleType>
  </xs:union>
 </xs:simpleType>
 <xs:simpleType name="DayType">
  <xs:restriction base="xs:NMTOKEN">
   <xs:pattern value="[m|M][o|O]"/>
   <xs:pattern value="[t|T][u|U]"/>
   <xs:pattern value="[w|W][e|E]"/>
   <xs:pattern value="[t|T][h|H]"/>
   <xs:pattern value="[f|F][r|R]"/>
   <xs:pattern value="[s|S][a|A]"/>
   <xs:pattern value="[s|S][u|U]"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:complexType name="TimeType">
  <xs:annotation>
<xs:documentation>Exactement un des deux attributs "dtend" et "duration" doit apparaître. Aucun des attributs suivant
freq n'a de signification si freq n'apparaît pas. </xs:documentation>
  </xs:annotation>
  <xs:group ref="Node"/>
  <xs:attribute name="dtstart" type="xs:string" use="required">
   <xs:annotation>
<xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
   </xs:annotation>
  </xs:attribute>
  <xs:attribute name="dtend" type="xs:string" use="optional">
   <xs:annotation>
<xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
   </xs:annotation>
  </xs:attribute>
  <xs:attribute name="duration" type="xs:string" use="optional">
   <xs:annotation>
<xs:documentation>RFC 2445 DURATION</xs:documentation>
   </xs:annotation>
  </xs:attribute>
  <xs:attribute name="freq" type="FreqType" use="optional"/>
  <xs:attribute name="interval" type="xs:positiveInteger"
```

```
      default="1"/>
    <xs:attribute name="until" type="xs:string" use="optional">
      <xs:annotation>
 <xs:documentation>RFC 2445 DATE-TIME</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="count" type="xs:positiveInteger"
      use="optional"/>
    <xs:attribute name="bysecond" type="xs:string" use="optional">
      <xs:annotation>
<xs:documentation>Liste séparée par des virgules des secondes dans une minute. Les valeurs valides sont de 0 à 59.
</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="byminute" type="xs:string" use="optional">
      <xs:annotation>
<xs:documentation>Liste séparée par des virgules des minutes au sein d'une heure. Les valeurs valides sont de 0 à 59.
</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="byhour" type="xs:string" use="optional">
      <xs:annotation>
<xs:documentation>Liste séparée par des virgules des heures du jour. Les valeurs valides sont de 0 à 23.
/xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="byday" type="xs:string" use="optional">
      <xs:annotation>
<xs:documentation>Liste séparée par des virgules des jours de la semaine. Les valeurs valides sont "MO", "TU", "WE",
"TH", "FR", "SA" et "SU". Ces valeurs ne sont pas sensibles à la casse. Chacune peut être précédée par un entier positif
(+n) ou négatif (-n).</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="bymonthday" type="xs:string" use="optional">
      <xs:annotation>
<xs:documentation>Liste séparée par des virgules des jours du mois. Les valeurs valides sont de 1 à 31 ou de -31 à -1.
</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="byyearday" type="xs:string" use="optional">
      <xs:annotation>
<xs:documentation>Liste séparée par des virgules des jours de l'année. Les valeurs valides sont de 1 à 366 ou de -366 à
-1.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="byweekno" type="xs:string" use="optional">
      <xs:annotation>
<xs:documentation>Liste séparée par des virgules des ordinaux spécifiant les semaines de l'année. Les valeurs valides sont
de 1 à 53 ou de -53 à -1.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="bymonth" type="xs:string" use="optional">
      <xs:annotation>
<xs:documentation>Liste séparée par des virgules des mois de l’année. Les valeurs valides sont de 1 à 12.
</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="wkst" type="DayType" default="MO"/>
    <xs:attribute name="bysetpos" type="YearDayType"/>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
  <xs:simpleType name="TZIDType">
    <xs:restriction base="xs:string"/>
```

```
  </xs:simpleType>
  <xs:simpleType name="TZURLType">
    <xs:restriction base="xs:anyURI"/>
  </xs:simpleType>
  <xs:complexType name="TimeSwitchType">
    <xs:complexContent>
      <xs:extension base="SwitchType">
        <xs:sequence>
          <xs:element name="time" type="TimeType" minOccurs="0"
            maxOccurs="unbounded"/>
          <xs:sequence minOccurs="0">
            <xs:element name="not-present" type="NotPresentAction"/>
            <xs:element name="time" type="TimeType" minOccurs="0"
              maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:element name="otherwise" type="OtherwiseAction"
            minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="tzid" type="TZIDType"/>
        <xs:attribute name="tzurl" type="TZURLType"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="time-switch" type="TimeSwitchType"
     substitutionGroup="switch"/>
  <xs:simpleType name="PriorityValues">
      <xs:restriction base="xs:NMTOKEN">
        <xs:pattern
           value="[e|E][m|M][e|E][r|R][g|G][e|E][n|N][c|C][y|Y]"/>
        <xs:pattern value="[u|U][r|R][g|G][e|E][n|N][t|T]"/>
      <xs:pattern value="[n|N][o|O][r|R][m|M][a|A][l|L]"/>
      <xs:pattern
         value="[n|N][o|O][n|N]-[u|U][r|R][g|G][e|E][n|N][t|T]"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="PriorityType">
    <xs:annotation>
<xs:documentation>Exactement un des trois attributs doit apparaître </xs:documentation>
    </xs:annotation>
    <xs:group ref="Node"/>
    <xs:attribute name="less" type="PriorityValues"/>
    <xs:attribute name="greater" type="PriorityValues"/>
    <xs:attribute name="equal" type="xs:string">
      <xs:annotation>
<xs:documentation>insensible à la casse</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>
  <xs:complexType name="PrioritySwitchType">
    <xs:complexContent>
      <xs:extension base="SwitchType">
        <xs:sequence>
          <xs:element name="priority" type="PriorityType"
            minOccurs="0" maxOccurs="unbounded"/>
          <xs:sequence minOccurs="0">
            <xs:element name="not-present" type="NotPresentAction"/>
            <xs:element name="priority" type="PriorityType"
              minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:element name="otherwise" type="OtherwiseAction"
            minOccurs="0"/>
        </xs:sequence>
```

```
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="priority-switch" type="PrioritySwitchType"
   substitutionGroup="switch"/>
<xs:simpleType name="LocationPriorityType">
  <xs:restriction base="xs:float">
    <xs:minInclusive value="0.0"/>
    <xs:maxInclusive value="1.0"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="LocationType">
  <xs:complexContent>
    <xs:extension base="ModifierType">
      <xs:group ref="Node"/>
      <xs:attribute name="url" type="xs:anyURI" use="required"/>
      <xs:attribute name="priority" type="LocationPriorityType"
         use="optional" default="1.0"/>
      <xs:attribute name="clear" type="YesNoType" default="no"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="LookupType">
  <xs:complexContent>
    <xs:extension base="ModifierType">
      <xs:all>
        <xs:element name="success" minOccurs="0">
          <xs:complexType>
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="notfound" minOccurs="0">
          <xs:complexType>
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="failure" minOccurs="0">
          <xs:complexType>
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
      </xs:all>
      <xs:attribute name="source" type="xs:string"
         use="required"/>
      <xs:attribute name="timeout" type="xs:positiveInteger"
         default="30"/>
      <xs:attribute name="clear" type="YesNoType" default="no"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="RemoveLocationType">
  <xs:complexContent>
    <xs:extension base="ModifierType">
      <xs:group ref="Node"/>
      <xs:attribute name="location" type="xs:string"
         use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="LogAction">
  <xs:complexContent>
    <xs:extension base="ActionType">
      <xs:group ref="Node"/>
```

```
      <xs:attribute name="name" type="xs:string" use="optional"/>
      <xs:attribute name="comment" type="xs:string"
        use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="log" type="LogAction"
  substitutionGroup="action"/>
<xs:complexType name="IncomingType">
  <xs:complexContent>
    <xs:extension base="TopLevelActionType"/>
  </xs:complexContent>
</xs:complexType>
<xs:element name="incoming" type="IncomingType"
  substitutionGroup="toplevelaction"/>
<xs:complexType name="OutgoingType">
  <xs:complexContent>
    <xs:extension base="TopLevelActionType"/>
  </xs:complexContent>
</xs:complexType>
<xs:element name="outgoing" type="OutgoingType"
  substitutionGroup="toplevelaction"/>
<xs:complexType name="ProxyAction">
  <xs:complexContent>
    <xs:extension base="ActionType">
      <xs:all>
        <xs:element name="busy" minOccurs="0">
          <xs:complexType>
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="noanswer" minOccurs="0">
          <xs:complexType>
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="failure" minOccurs="0">
          <xs:complexType>
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="redirection" minOccurs="0">
          <xs:complexType>
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="default" minOccurs="0">
          <xs:complexType>
            <xs:group ref="Node"/>
          </xs:complexType>
        </xs:element>
      </xs:all>
      <xs:attribute name="timeout" type="xs:positiveInteger"
        use="optional" default="20"/>
      <xs:attribute name="recurse" type="YesNoType"
        use="optional" default="yes"/>
      <xs:attribute name="ordering" type="OrderingType"
        use="optional" default="parallel"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="proxy" type="ProxyAction"
  substitutionGroup="action"/>
```

```
 <xs:complexType name="RedirectAction">
  <xs:complexContent>
   <xs:extension base="ActionType">
    <xs:attribute name="permanent" type="YesNoType"
     default="no"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:element name="redirect" type="RedirectAction"
   substitutionGroup="action"/>
 <xs:complexType name="RejectAction">
  <xs:complexContent>
   <xs:extension base="ActionType">
    <xs:attribute name="status" type="StatusType"
     use="required"/>
    <xs:attribute name="reason" type="xs:string"
     use="optional"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:element name="reject" type="RejectAction"
   substitutionGroup="action"/>
 <xs:complexType name="MailAction">
  <xs:complexContent>
   <xs:extension base="ActionType">
    <xs:group ref="Node"/>
    <xs:attribute name="url" type="xs:anyURI" use="required"/>
   </xs:extension>
  </xs:complexContent>
 </xs:complexType>
 <xs:element name="mail" type="MailAction"
   substitutionGroup="action"/>
 <xs:complexType name="SubAction">
  <xs:attribute name="ref" type="xs:string" use="required"/>
 </xs:complexType>
 <xs:complexType name="AncillaryType"/>
 <xs:complexType name="SubactionType">
  <xs:group ref="Node"/>
  <xs:attribute name="id" use="required"/>
 </xs:complexType>
 <xs:complexType name="CPLType">
   <xs:sequence>
   <xs:element name="ancillary" type="AncillaryType" minOccurs="0"
     maxOccurs="1"/>
   <xs:element name="subaction" type="SubactionType" minOccurs="0"
     maxOccurs="unbounded"/>
   <xs:element ref="toplevelaction" minOccurs="0"
     maxOccurs="unbounded">
    <xs:annotation>
<xs:documentation>Aucune action de niveau supérieur NE DOIT apparaître plus d'une fois.</xs:documentation>
    </xs:annotation>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:element name="cpl" type="CPLType"/>
</xs:schema>
END
```

Références

[H.323] Union Internationale des Télécommunications – Secteur de la normalisation des télécommunications, Recommandation UIT-T H.323, "Systèmes de communication multimédia à base de paquets", Genève, Suisse, juillet 2003.

[HTML] Raggett, D., Le Hors, A., et I. Jacobs, "HTML 4.01 Specification", W3C Recommendation REC-html401-19991224, World Wide Web Consortium (W3C), décembre 1999. Disponible à http://www.w3.org/TR/html4/

[ISO8601] ISO (Organisation internationale de normalisation), "Format d'éléments et d'échange de données – Échanges d'information – Représentation des dates et des heures", Norme ISO 8601:2000(E), décembre 2000.

[ISO8879] ISO (Organisation internationale de normalisation), "Traitement de l'information – Systèmes de texte et de bureau – Langage de balisage standard généralisé (SGML)", Norme ISO 8879:1986(E), octobre 1986.

[NomsXML] Bray, T., Hollander, D., et A. Layman, "Namespaces in XML", W3C Recommendation REC-xml-names-19990114, World Wide Web Consortium (W3C), janvier 1999. Disponible à http://www.w3.org/TR/REC-xml-names/ .

[Q.931] Union Internationale des Télécommunications – Secteur de la normalisation des télécommunications, "Système de signalisation d'abonné numérique n° 1 (DSS 1) – Spécification de la couche 3 d'interface usager-réseau RNIS pour la commande d'appel de base", Recommandation UIT-T Q.931, Genève, Suisse, mars 1993.

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

[RFC2141] R. Moats, "Syntaxe des URN", mai 1997.

[RFC2445] F. Dawson et D. Stenerson, "Spécification centrale des objets de calendrier et de programmation de l'Internet (iCalendar)", novembre 1998. *(P.S.)*

[RFC2483] M. Mealling et R. Daniel, "Services de résolution d'URI nécessaires pour la résolution d'URN", janvier 1999.

[RFC2648] R. Moats, "Espace de nom d'URN pour les documents de l'IETF", août 1999. *(Information)*

[RFC2824] J. Lennox et H. Schulzrinne, "Cadres et exigences du langage de traitement d'appel (CPL)", mai 2000.

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

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

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

[RFC3261] J. Rosenberg et autres, "SIP : Protocole d'initialisation de session", juin 2002. (*Mise à jour par* RFC3265, RFC3853, RFC4320, RFC4916, RFC5393, RFC6665)

[RFC3508] O. Levin, "Enregistrement du schéma des adresses universelles (URL) de la Recommandation UIT-T H.323", avril 2003.

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

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

[Sources] Eggert, P., "Sources for Time Zone et Daylight Saving Time Data". Disponible à http://www.twinsun.com/tz/tz-link.htm .

[TRANSP] Davis, M. F., "Case Mappings", Unicode Standard Annex #21, Unicode Consortium, mars 2001. Révision 5 ; partie 3.2.0 de Unicode. Disponible à http://www.unicode.org/unicode/reports/tr21/ .

[UNICODE] Davis, M. F. et M. Duerst, "Unicode Normalization Forms", Unicode Standard Annex #15, Unicode Consortium, avril 2003. Révision 23 ; partie de Unicode 4.0.0. Disponible à http://www.unicode.org/unicode/reports/tr15/ .

[XLink] DeRose, S., Maler, E., Orchard, D., et B. Trafford, "XML Linking Language (XLink) Version 1.0", W3C Recommendation REC-xlink-20010627, World Wide Web Consortium (W3C), juin 2001. Disponible à http://www.w3.org/TR/xlink/ .

[XML] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., et F. Yergeau, "Extensible Markup Language (XML) 1.0 (3è édition)", W3C Recommendation REC-xml-20040204, World Wide Web Consortium (W3C), février 2004. Disponible à http://www.w3.org/XML/ .

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

Xiaotao Wu
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
mél : xiaotaow@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

Déclaration complète de droits de reproduction

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