UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

H.248.1 v2

Corrigendum 1

(03/2004)

SÉRIE H: SYSTÈMES AUDIOVISUELS ET MULTIMÉDIAS

Infrastructure des services audiovisuels – Procédures de communication

---

Protocole de commande de passerelle: version 2

Corrigendum 1

Recommandation UIT-T H.248.1 v2 (2002) – Corrigendum 1

RECOMMANDATIONS UIT-T DE LA SÉRIE H

SYSTÈMES AUDIOVISUELS ET MULTIMÉDIAS

CARACTÉRISTIQUES DES SYSTÈMES VISIOPHONIQUES	H.100–H.199
INFRASTRUCTURE DES SERVICES AUDIOVISUELS	
Généralités	H.200–H.219
Multiplexage et synchronisation en transmission	H.220–H.229
Aspects système	H.230–H.239
Procédures de communication	**H.240–H.259**
Codage des images vidéo animées	H.260–H.279
Aspects liés aux systèmes	H.280–H.299
Systèmes et équipements terminaux pour les services audiovisuels	H.300–H.349
Architecture des services d'annuaire pour les services audiovisuels et multimédias	H.350–H.359
Architecture de la qualité de service pour les services audiovisuels et multimédias	H.360–H.369
Services complémentaires en multimédia	H.450–H.499
PROCÉDURES DE MOBILITÉ ET DE COLLABORATION	
Aperçu général de la mobilité et de la collaboration, définitions, protocoles et procédures	H.500–H.509
Mobilité pour les systèmes et services multimédias de la série H	H.510–H.519
Applications et services de collaboration multimédia mobile	H.520–H.529
Sécurité pour les systèmes et services multimédias mobiles	H.530–H.539
Sécurité pour les applications et services de collaboration multimédia mobile	H.540–H.549
Procédures d'interfonctionnement de la mobilité	H.550–H.559
Procédures d'interfonctionnement de collaboration multimédia mobile	H.560–H.569
SERVICES À LARGE BANDE ET MULTIMÉDIAS TRI-SERVICES	
Services multimédias à large bande sur VDSL	H.610–H.619

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T H.248.1 v2

Protocole de commande de passerelle: version 2

Corrigendum 1

Résumé

Aux fins d'une meilleure évolutivité, la présente Recommandation décompose en sous éléments fonctionnels la fonction de passerelle H.323 définie dans la Rec. UIT-T H.246. Elle spécifie les protocoles utilisés par ces éléments pour communiquer, ce qui permet aux implémentations de passerelles H.323 d'avoir une évolutivité élevée et qui incite à la multiplication de capacités, telles que les commutateurs SS7, de réseau à commutation de circuits (RCC). Ces sous éléments permettent également aux passerelles H.323 d'être constituées d'éléments issus de multiples vendeurs répartis sur de multiples plates-formes physiques. L'objet de la présente Recommandation est d'ajouter les capacités actuellement définies pour les systèmes H.323, en vue d'offrir de nouveaux moyens d'effectuer les opérations déjà assurées dans les systèmes H.323.

La présente Recommandation apporte plusieurs améliorations à la version 1 de la Rec. UIT-T H.248.1:

– analyse des propriétés, des signaux, des événements et des statistiques distincts;

– traitement multiplex amélioré;

– topologie des flux;

– description améliorée des profils;

– modification de la capacité serviceChange.

Le Corrigendum 1 permet de corriger plusieurs erreurs décelées dans la Rec. UIT-T H.248.1, notamment en ce qui concerne les points suivants:

• spécification des types applicables aux propriétés rtp/jit et rtp/delay dans le § E.12.4;

• définition du symbole '#' dans la structure "unequal" du codage alphanumérique;

• définition du symbole représentant le contexte NULL dans le codage alphanumérique;

• corrections relatives: à des exemples de statistiques dans l'Appendice I, à des lignes directrices relatives au paquetage de statistiques dans le § 12.1.5; au renvoi relatif à une analyse individuelle ou à une analyse ambiguë; au renvoi relatif à une analyse de contexte; à une erreur de typographie dans le § 7.1.2;

• spécification de la signification de "automatique" au § E.13 ("Paquetage de circuit TDM") de l'Annexe E;

• adjonction de code et d'une valeur binaire concernant un temps de mise en paquets dans l'Annexe C;

• clarification des principes de remplacement par des caractères génériques et du remplacement par des caractères génériques dans le descripteur Topology;

• clarification concernant: les statistiques et la commande Move; la modification de terminaisons par des contrôleurs de passerelle média; des commandes facultatives dans une action; l'ordre des transactions; la priorité de la propriété Mode du descripteur LocalControl par rapport au mode SDP; le traitement numérique; l'utilisation des symboles de temporisateurs de script de numérotation avec intervalle de notation; l'utilisation de l'identificateur de flux StreamID = 0; le renvoi de la commande AuditCapabilities pour des

valeurs de chaîne; la négociation de la version du protocole;

- clarification du fait que le paquetage de réseau peut s'appliquer à un réseau TDM.

NOTE – La présente Recommandation comporte la nouvelle numérotation de la Rec. UIT-T H.248, ainsi que les Annexes A à E et l'Appendice I de celle-ci.

Source

Le Corrigendum 1 de la Recommandation H.248.1 v2 (2002) de l'UIT-T a été approuvé le 15 mars 2004 par la Commission d'études 16 (2001-2004) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT avait été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

		Page
1)	Paragraphe 2.1	1
2)	Paragraphe 6.2	1
3)	Paragraphe 6.3	3
4)	Paragraphe 7, Commandes	5
5)	Paragraphe 8	12
6)	Paragraphe 11.3	12
7)	Paragraphe 12	13
8)	Annexe B Codage alphanumérique du protocole	14
9)	Annexe C Etiquettes des propriétés de flux média	26
10)	Annexe E Paquetages de base	27
11)	Appendice I Exemples de flux de communication	28

Recommandation UIT-T H.248.1 v2

Protocole de commande de passerelle: version 2

Corrigendum 1

1) Paragraphe 2.1

Modifier le § 2.1 comme suit:

2.1 Références normatives

– Recommandation UIT-T H.225.0 (2000), *Protocoles de signalisation d'appel et paquétisation des flux monomédias dans les systèmes de communication multimédias en mode paquet.*

– Recommandation UIT-T H.235 (2000), *Sécurité et cryptage des terminaux multimédias de la série H (terminaux H.323 et autres terminaux de type H.245)*.

– Recommandation UIT-T H.245 (2001), *Protocole de commande pour communications multimédias*.

– Recommandation UIT-T H.246 (1998), *Interfonctionnement des terminaux multimédias de la série H avec d'autres terminaux multimédias de la série H et des terminaux vocaux ou en bande vocale sur le RTGC et le RNIS.*

– Recommandation UIT-T H.248.4 (2000), *Protocole de commande de passerelle: transport par le protocole de transmission de commande de flux (SCTP)*, plus Corr.1 (2004).

– Recommandation UIT-T H.248.5 (2000), *Protocole de commande de passerelle: transport sur réseau ATM.*

– Recommandation UIT-T H.248.8 (2002), *Protocole de commande de passerelle: description des codes d'erreur et des causes de changement de service*, plus Amendement 1 (2004).

•••••

2) Paragraphe 6.2

Modifier le § 6.2 comme suit:

6.2 Terminaisons

Les terminaisons peuvent subir l'application de signaux (se reporter au § 7.1.11). Les terminaisons peuvent être programmées de façon à détecter des événements, dont l'apparition peut déclencher l'envoi de messages de notification vers le contrôleur MGC ou déclencher une action de la passerelle MG. Des statistiques peuvent être cumulées au sujet d'une terminaison. Ces statistiques sont signalées au contrôleur MGC sur demande (au moyen d'une commande AuditValue, se reporter au § 7.2.5) et lorsque la terminaison ~~est retirée d'un contexte~~cesse d'exister ou est renvoyée au contexte NULL en raison de l'application d'une commande Substract.

•••••

6.2.1 Dynamique de terminaison

Le protocole peut être utilisé pour créer de nouvelles terminaisons et pour modifier les valeurs des propriétés de terminaisons existantes. Ces modifications comprennent la possibilité d'ajouter ou de supprimer des événements et/ou des signaux. Les propriétés des terminaisons ainsi que les

événements et les signaux sont décrits dans les paragraphes qui suivent. Un contrôleur MGC ne peut libérer/modifier que les terminaisons (et les ressources qu'elles représentent) qui se trouvent dans le contexte NULL ou qui ont été préalablement saisies au moyen, par exemple, de la commande Add.

6.2.2 TerminationIDs (identificateurs de terminaison)

Les terminaisons sont désignées par un identificateur de terminaison qui est une séquence arbitraire, choisie par la passerelle MG.

Les identificateurs de terminaisons physiques sont fournis dans la passerelle MG. Ils peuvent être choisis de façon à posséder une structure. Par exemple, un identificateur de terminaison peut se composer d'un faisceau de circuits et d'une jonction locale dans ce faisceau.

Un mécanisme de remplacement par des caractères génériques, utilisant deux types de caractère générique, peut être utilisé avec les identificateurs de terminaison. Ces deux caractères sont ALL et CHOOSE. Le premier sert à désigner simultanément plusieurs terminaisons tandis que le second sert à indiquer à une passerelle MG qu'elle doit sélectionner une terminaison correspondant à l'identificateur de terminaison partiellement spécifié. Cela permet à un contrôleur MGC de demander, par exemple, à une passerelle MG de choisir un circuit dans un faisceau de circuits.

~~Si le caractère ALL est utilisé dans l'identificateur de terminaison d'une commande, l'effet est identique à une répétition de la commande avec chacun des identificateurs de terminaison réels qui correspondent. L'emploi de ALL n'inclut pas la terminaison ROOT. Etant donné que chacune de ces commandes peut générer une réponse, la taille de la réponse complète peut être importante. Si des réponses individuelles ne sont pas requises, une réponse générique peut être demandée. Dans ce cas, une seule réponse est générée et elle contient l'UNION de toutes les réponses individuelles qui auraient été autrement générées, les valeurs répétées étant supprimées. Par exemple, étant donné une terminaison Ta dont les propriétés seraient p1 = a, p2 = b et une terminaison Tb dont les propriétés seraient p2 = c, p3 = d, une réponse UNION contiendrait un identificateur de terminaison remplacé par un caractère générique et la séquence de propriétés p1 = a, p2 = b,c et p3 = d. La réponse générique peut être particulièrement utile dans les commandes d'Audit.~~

~~Le codage du mécanisme de remplacement par des caractères génériques est détaillé dans les Annexes A et B.~~

6.2.3 Paquetages

Différents types de passerelle peuvent implémenter des terminaisons possédant des caractéristiques très différentes. Le protocole tient compte des variantes de terminaison en permettant que leurs implémentations par des passerelles MG aient des propriétés, des événements, des signaux et des statistiques de type facultatif.

•••••

6.2.4 Propriétés et descripteurs de terminaison

•••••

Le tableau ci-dessous énumère tous les descripteurs possibles ainsi que leur usage. Tous les descripteurs ne sont pas autorisés en tant que paramètres d'entrée ou de sortie d'une commande donnée.

Nom du descripteur	Description
Modem	Désigne, le cas échéant, le type et les propriétés d'un modem (Note).

•••••

Erreur	Contient un code d'erreur et éventuellement un texte d'erreur; il peut figurer dans les réponses aux commandes et dans les demandes de notification.
NOTE – Le descripteur ModemDescriptor a été désapprouvé dans la Rec. UIT-T H.248.1 version ~~1~~ 2 (~~03~~05/2002).	

6.2.5 Terminaison racine

•••••

3) Paragraphe 6.3

Ajouter le § 6.3 comme suit:

6.3 Principes de remplacement par des caractères génériques

Le présent paragraphe spécifie les mécanismes applicables à toutes les commandes spécifiant des identificateurs de contexte et/ou de terminaison comprenant des caractères génériques. Lorsque l'on traite ces commandes, deux types de remplacement par des caractères génériques doivent être considérés:

1) remplacement par des caractères génériques dans un identificateur de contexte;

2) remplacement par des caractères génériques dans un identificateur de terminaison.

Lorsque l'on effectue une transaction spécifiant des identificateurs de contexte et éventuellement des identificateurs de terminaison comprenant des caractères génériques, toutes les commandes de la transaction sont exécutées séquentiellement pour une instance particulière de l'identificateur de contexte avant de passer à l'instance suivante de cet identificateur. Lorsqu'une transaction comprend plusieurs commandes, il faut que l'identificateur de terminaison (générique ou spécifique) spécifié dans la première commande corresponde à une instance spécifique d'un identificateur de contexte pour que les commandes suivantes puissent être exécutées. Si un identificateur de terminaison (générique ou spécifique) spécifié dans la ou les commandes suivantes de cette transaction ne correspond pas à cette instance spécifique de l'identificateur de contexte, le code d'erreur 431 est renvoyé et le traitement des instances suivantes de l'identificateur de contexte générique est arrêté, sauf si la commande ayant généré l'erreur est signalée comme étant facultative.

L'exécution de commandes spécifiant des combinaisons particulières d'identificateurs comprenant des caractères génériques est étudiée ci-dessous.

6.3.1 Identificateur de contexte spécifique et identificateur de terminaison générique

Lorsque l'identificateur de contexte est spécifique et que le caractère générique ALL est utilisé dans l'identificateur de terminaison d'une commande, l'effet est identique à une répétition de la commande pour chacun des identificateurs de terminaison qui correspondent. L'utilisation de ALL ne concerne pas la terminaison ROOT. Etant donné que chacune de ces commandes peut générer une réponse, la taille de la réponse complète peut être importante. Ainsi, si le caractère générique concorde avec plusieurs identificateurs de terminaison dans le contexte considéré, toutes les correspondances possibles seront essayées et les résultats seront signalés à chaque fois. Si aucune des terminaisons spécifiées par l'identificateur de terminaison générique ne se trouve dans le contexte spécifique considéré, le code d'erreur 431 est renvoyé. Aucune erreur n'est renvoyée pour les terminaisons spécifiées par l'identificateur de terminaison générique qui ne se trouvent pas dans le contexte spécifique considéré.

Supposons par exemple qu'une passerelle comprenne quatre terminaisons t1/1, t1/2, t2/1 et t2/2. On suppose que t1/1 et t2/1 appartiennent au contexte 1 et que t1/2 et t2/2 appartiennent au contexte 2.

La commande:

Context=1{Command=t1/{Descriptor/s}}*

renvoie:

Context=1{Command=t1/1{Descriptor/s}}

6.3.2 Identificateur de contexte générique (ALL) et identificateur de terminaison spécifique

Lorsque l'identificateur de contexte est générique (c'est-à-dire identificateur de contexte = ALL) et que l'identificateur de terminaison est complètement spécifié, l'effet est identique à celui produit par une commande spécifiant un contexte non NULL comprenant la terminaison spécifiée. Une recherche doit donc être menée pour trouver le contexte approprié et seule une instance de la commande est exécutée. Aucune erreur n'est signalée pour les contextes ne comprenant pas la terminaison spécifiée. Si la terminaison ne figure dans aucun contexte (non NULL), l'erreur 431 est renvoyée. Utiliser ce type d'action plutôt que spécifier l'identificateur de contexte est déconseillé mais peut être utile, par exemple pour remédier à un conflit d'états entre la passerelle média et le contrôleur de passerelle média.

Par exemple, en reprenant la configuration de passerelle décrite plus haut.

La commande:

Context={Command=t1/1{Descriptor/s}}*

renvoie:

Context=1{Command=t1/1{Descriptor/s}}

6.3.3 Identificateur de contexte générique (ALL) et identificateur de terminaison générique

Lorsque l'identificateur de contexte est générique (c'est-à-dire identificateur de contexte = ALL) et que l'identificateur de terminaison est générique, l'effet est identifique à celui produit par la répétition de la commande pour chacun des identificateurs de terminaison spécifiés par l'identificateur de terminaison générique dans chaque contexte non NULL comprenant un ou plusieurs de ces identificateurs de terminaison. Ainsi, si le caractère générique concorde avec plusieurs identificateurs de terminaison d'une instance spécifique de l'identificateur de contexte générique, toutes les correspondances possibles seront essayées et les résultats seront signalés à chaque fois. Aucune erreur n'est renvoyée pour les contextes ne contenant pas une terminaison correspondant à l'identité de terminaison générique. Aucun code d'erreur n'est renvoyé pour les terminaisons spécifiées par l'identificateur de terminaison générique qui ne se trouvent pas dans une instance spécifique de l'identificateur de contexte générique. S'il n'existe pas de correspondance entre l'identificateur de contexte générique et l'identificateur de terminaison générique, l'erreur 431 est renvoyée.

Par exemple, en reprenant la configuration de passerelle décrite plus haut,

La commande:

Context={Command=t1/*{Descriptor/s}}*

renvoie:

Context=1{Command=t1/1{Descriptor/s}}

Context=2{Command=t1/2{Descriptor/s}}

Lorsque plusieurs commandes figurent dans une demande spécifiant un identificateur de terminaison générique et/ou un identificateur de contexte générique, alors, si la première commande ne parvient pas à faire correspondre la première instance associant un identificateur de contexte et un identificateur de terminaison, la commande suivante de la demande ne sera pas exécutée pour cette instance.

6.3.4 Réponses génériques

Si des réponses individuelles ne sont pas requises, une réponse générique peut être demandée. Dans un tel cas, une seule réponse est générée et elle contient l'UNION de toutes les réponses individuelles qui auraient été en principe générées, les répétitions étant supprimées. Par exemple, pour une terminaison Ta ayant pour propriétés p1 = a, p2 = b et une terminaison Tb ayant pour propriétés p2 = c, p3 = d, une réponse UNION contiendrait un identificateur de terminaison générique ainsi que la séquence de propriétés p1 = a, p2 = b,c et p3 = d. Faire appel à une réponse générique peut être particulièrement utile dans le cas d'une commande Audit. Si l'on utilise une réponse UNION générique associée à un contexte générique, une seule réponse est envoyée et elle comprend l'UNION des différentes terminaisons spécifiées par l'identificateur de terminaison. Une telle réponse contiendrait le paramètre Context = ALL, une identité de terminaison générique et la séquence de propriétés appropriée.

Si une erreur se produit au cours de l'exécution d'une requête générique spécifiant une réponse générique, un traitement particulier est nécessaire pour fournir des informations utiles concernant les éventuelles erreurs tout en conservant une réponse de taille modeste. Lorsqu'une réponse générique est requise, toutes les instances (telles que spécifiées plus haut) de la commande considérée doivent être exécutées même si l'une ou plusieurs d'entre elles conduisent à des erreurs; toutefois, les commandes suivantes de la transaction ne seront pas exécutées (sauf spécification du caractère facultatif de la commande). Plusieurs réponses de commande doivent être renvoyées pour la commande ayant généré une erreur. La première réponse de commande doit être la réponse générique normale comprenant l'UNION des réponses aux commandes dont l'exécution a été réussie. Si aucune commande n'a pu être exécutée correctement, le paramètre UNION doit être vide. D'autres réponses de commandes relatives aux différents identificateurs de transaction ayant échouée doivent être renvoyées accompagnés du descripteur d'erreur appropriée.

Par exemple

La commande:

Context={Command=t1/*{Descriptor/s}}*

renvoie en cas d'erreur:

Context={Command=t1/*{Union response descriptors},*

Command=t1/3{Error=errorcode}}

Le codage du mécanisme de remplacement par des caractères génériques est détaillé dans les Annexes A et B.

•••••

4) Paragraphe 7, Commandes

a) Modifier le § 7.1 comme suit:

7.1.2 Descripteur Modem

Le descripteur Modem spécifie le type et les paramètres d'un modem, éventuellement nécessaires pour utilisation lors d'une conversation en mode H.324 et en mode texte. Ce descripteur désigne les types de modem suivants: V.18, V.22, V.22 *bis*, V.32, V.32 *bis*, V.34, V.90, V.91, RNIS synchrone, avec possibilité d'extensions. Par défaut, aucun descripteur Modem n'est présent dans une terminaison.

L'emploi du descripteur ModemDescriptor est désapprouvé dans la Rec. UIT-T H.248.1 version ~~1~~ (~~03~~2(05/2002) et dans les versions ultérieures. Cela veut dire que ce descripteur ne fera pas partie du contenu transmis, et, s'il est reçu, il sera soit ignoré soit traité en fonction de l'implémentation. Le

type de modem est à spécifier comme attribut des flux de données dans les descripteurs LocalDescriptor et RemoteDescriptor.

•••••

7.1.4 Descripteur Media

Le descripteur Media spécifie les paramètres de tous les flux médias. Ces paramètres sont structurés en deux descripteurs: un descripteur TerminationState qui spécifie les propriétés d'une terminaison qui ne dépendent pas du flux et un ou plusieurs descripteurs Stream dont chacun décrit un seul flux média.

Un flux est identifié par un identificateur Stream (compris entre 1 et 65535) servant à associer ce flux au contexte auquel il appartient. Plusieurs flux sortant d'une terminaison doivent être synchronisés les uns avec les autres. A l'intérieur du descripteur Stream, il existe trois sous-descripteurs, désignés par LocalControl, Local et Remote. La relation entre ces descripteurs est donc la suivante:

•••••

7.1.7 Descripteur LocalControl

Le descripteur LocalControl contient la propriété Mode, les propriétés ReserveGroup et ReserveValue ainsi que les propriétés d'une terminaison (définies dans des paquetages) qui sont spécifiques au flux et présentent un intérêt entre la passerelle MG et le contrôleur MGC. Les valeurs des propriétés peuvent être spécifiées comme indiqué au § 7.1.1.

Les valeurs autorisées pour la propriété Mode sont: "send-only" (émission seulement), "receive-only" (réception seulement), "send/receive" (émission/réception), "inactive" (inactive) et "loop-back" (bouclage). Les valeurs "send" et "receive" se rapportent à l'intérieur d'un contexte si bien que, par exemple, un flux réglé sur "mode = sendOnly" ne transmettra pas au contexte le média reçu. La valeur par défaut de la propriété de mode est "inactive". Les signaux et les événements ne sont pas affectés par le paramètre de mode. La propriété Mode du descripteur LocalControl est prioritaire vis-à-vis de n'importe quel mode spécifié dans les descripteurs Local et Remote.

•••••

7.1.14.2 Temporisateurs de script de numérotation

Le paquetage des chiffres conformes à un script de numérotation peut être protégé par trois temporisateurs, à savoir un temporisateur de départ (T), un temporisateur court (S) et un temporisateur long (L), comme suit:

1) le temporisateur de départ (T) est utilisé avant ~~que des chiffres aient été composés~~tout chiffre du script de numérotation disponible pour être traité. Si la valeur du temporisateur de départ est supplantée par une valeur fixée à zéro (T = 0), le temporisateur de départ doit être désactivé. Cela implique que la passerelle MG attendra indéfiniment les chiffres.

•••••

7.1.14.3 Syntaxe de script de numérotation

•••••

En plus de ces symboles d'événement, la chaîne peut contenir les spécificateurs de temporisation inter-événements "S" et "L" ainsi que le modificateur de durée "Z". "S" et "L" indiquent respectivement qu'il convient que la passerelle MG utilise le temporisateur court (S) ou le temporisateur long (L) pour des événements subséquents, annulant ainsi les règles décrites précédemment. Si un spécificateur de temporisation explicite est en vigueur dans une séquence d'événements en variante, mais qu'aucun autre n'est fourni dans les autres variantes candidates, on doit utiliser la valeur de temporisation fixée par le spécificateur explicite. Si toutes les séquences

comportant des contrôles de temporisation explicites sont sorties du paquetage candidat, la temporisation reprend les règles par défaut fournies précédemment. S'ils sont utilisées dans un intervalle de notation, les spécificateurs S et L doivent être ignorés. Enfin, le temporisateur long sera employé si des spécificateurs de temporisation sont en conflit et en vigueur dans des séquences en variante différentes.

La lettre "Z" désigne un événement de longue durée: placée en face de symbole(s) désignant un ou des événements qui satisfont à la position de chiffre donnée, elle indique que la position n'est satisfaite que si la durée de l'événement ne dépasse pas le seuil de longue durée. La valeur de ce seuil est supposée fournie dans la passerelle MG, mais elle peut, à l'instar des temporisateurs T, L et S, être supplantée par une autre valeur spécifiée dans le script de numérotation. Si le spécificateur Z n'est pas suivi par un "chiffre" (0-9 ou A-K), la passerelle média doit rejeter le script de numérotation en tant que procédure non valide. Lorsqu'il est utilisé dans un intervalle de notation, le spécificateur Z s'applique uniquement au chiffre immédiatement suivant. Lorsqu'il est utilisé immédiatement avant un intervalle de notation, le modificateur Z s'applique à tous les chiffres de l'intervalle (ce qui signifie qu'un événement désigné dans l'intervalle doit être de longue durée).

7.1.14.4 Evénement d'achèvement de script de numérotation

•••••

7.1.15 Descripteur Statistics (de statistiques)

Le descripteur Statistics donne des informations décrivant l'état et l'usage d'une terminaison pendant son existence (terminaison éphémère) ou lorsqu'elle n'est pas dans le contexte NULL (terminaison physique)~~dans un contexte spécifique~~. Le cas échéant, il existe un paquetage de statistiques normalisées pour chaque terminaison (par exemple le nombre d'octets émis et reçus). Les propriétés statistiques particulières qui sont signalées pour une terminaison donnée sont déterminées par les paquetages réalisés par la terminaison. Par défaut, ces statistiques sont signalées lorsque la terminaison cesse d'exister ou est renvoyé au contexte NULL en raison de l'application d'une commande Substract~~ est soustraite du contexte par une commande Subtract~~. Ce comportement peut être annulé en incorporant un descripteur Audit vide dans la commande Substract. Des statistiques peuvent également être renvoyées par la commande AuditValue ou par toute commande Add/Move/Modify utilisant le descripteur Audit.

Les statistiques sont cumulatives. Le fait de les signaler ne les réinitialise pas. Les statistiques sont réinitialisées lorsqu'une terminaison cesse d'exister ou est renvoyé au contexte NULL en raison de l'application d'une commande Substract~~est soustraite d'un contexte~~.

7.1.16 Descripteurs de paquetages

•••••

7.1.18 Descripteur Topology

•••••

- le triplet (*T1*, *T2*, isolate) signifie que les terminaisons correspondant à *T2* ne reçoivent pas de média des terminaisons correspondant à *T1* et inversement;
- le triplet (*T1*, *T2*, oneway) signifie que les terminaisons correspondant à *T2* reçoivent des médias des terminaisons correspondant à *T1* mais pas inversement. Dans ce cas, l'utilisation du caractère générique ALL, de façon qu'il y ait des terminaisons correspondant ~~à la fois~~ à T1 ~~et~~ ou à T2 *mais pas à T1 et T2*, ~~n'~~est ~~pas~~ autorisée;
- le triplet (*T1*, *T2*, bothway) signifie que les terminaisons correspondant à *T2* reçoivent des médias des terminaisons correspondant à *T1* et inversement. Dans ce cas, il est permis d'utiliser des caractères génériques de façon qu'il ait des terminaisons correspondant à la fois à *T1* et à *T2*. Cependant, dans ce dernier cas, aucun bouclage n'est introduit.

•••••

b) Modifier le § 7.2 comme suit:

7.2.1 Commande Add

La commande Add ajoute une terminaison à un contexte.

Identificateur de terminaison
[,MediaDescriptor]
[,ModemDescriptor] (*)
[,MuxDescriptor]
[,EventsDescriptor]
[,SignalsDescriptor]
[,DigitMapDescriptor]
[,ObservedEventsDescriptor]
[,EventBufferDescriptor]
[,StatisticsDescriptor]
[,PackagesDescriptor]

Add(TerminationID
[, MediaDescriptor]
[, ModemDescriptor] (*)
[, MuxDescriptor]
[, EventsDescriptor]
[, EventBufferDescriptor]
[, SignalsDescriptor]
[, DigitMapDescriptor]
[, AuditDescriptor]
)

(*) Le descripteur ModemDescriptor a été désapprouvé dans la Rec. UIT-T H.248.1 version ~~1~~2 (0~~3~~5/2002).

•••••

7.2.2 Commande Modify

La commande Modify modifie les propriétés d'une terminaison.

Identificateur de terminaison
[,MediaDescriptor]
[,ModemDescriptor] (*)
[,MuxDescriptor]
[,EventsDescriptor]
[,SignalsDescriptor]
[,DigitMapDescriptor]
[,ObservedEventsDescriptor]
[,EventBufferDescriptor]
[,StatisticsDescriptor]

[,PackagesDescriptor]

Modify(TerminationID

[, MediaDescriptor]

[, ModemDescriptor] (*)

[, MuxDescriptor]

[, EventsDescriptor]

[, EventBufferDescriptor]

[, SignalsDescriptor]

[, DigitMapDescriptor]

[, AuditDescriptor]

)

(*) Le descripteur ModemDescriptor a été désapprouvé dans la Rec. UIT-T H.248.1 version ~~1~~ 2 (~~03~~5/2002).

•••••

7.2.3 Commande Subtract

La commande Subtract déconnecte une terminaison de son contexte et renvoie des statistiques sur la participation de cette terminaison dans le contexte.

Identificateur de terminaison

[,MediaDescriptor]

[,ModemDescriptor] (*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,DigitMapDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

[,PackagesDescriptor]

Subtract(TerminationID

[, AuditDescriptor]

)

(*) Le descripteur ModemDescriptor a été désapprouvé dans la Rec. UIT-T H.248.1 version ~~1~~ 2 (~~03~~5/2002).

•••••

7.2.4 Commande Move

La commande Move déplace une terminaison vers un autre contexte à partir de son contexte actuel, au cours d'une seule opération atomique. La commande Move est la seule commande qui se rapporte à une terminaison dans un contexte différent de celui auquel la commande s'applique. La commande Move ne doit pas être utilisée pour déplacer des terminaisons en direction de ou hors du contexte néant.

Identificateur de terminaison

[,MediaDescriptor]

[,ModemDescriptor] (*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,DigitMapDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

[,PackagesDescriptor]

Move(TerminationID

[, MediaDescriptor]

[, ModemDescriptor] (*)

[, MuxDescriptor]

[, EventsDescriptor]

[, EventBufferDescriptor]

[, SignalsDescriptor]

[, DigitMapDescriptor]

[, AuditDescriptor]

)

(*) Le descripteur ModemDescriptor a été désapprouvé dans la Rec. UIT-T H.248.1 version ~~1~~2 (0~~3~~5/2002).

•••••

7.2.5 Commande AuditValue

La commande AuditValue renvoie les valeurs actuelles des propriétés, des événements, des signaux et des statistiques associés aux terminaisons. Elle peut demander le contenu d'un descripteur ou d'une propriété, d'un événement, d'un signal ou d'une statistique.

Identificateur de terminaison

[,MediaDescriptor]

[,ModemDescriptor] (*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,DigitMapDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

[,PackagesDescriptor]

AuditValue(TerminationID,

AuditDescriptor

)

(*) Le descripteur ModemDescriptor a été désapprouvé dans la Rec. UIT-T H.248.1 version ~~1~~2 (0~~3~~5/2002).

•••••

7.2.6 Commande AuditCapabilities

La commande AuditCapabilities renvoie les valeurs possibles de propriétés, d'événements, de signaux et de statistiques associés à des terminaisons. Elle peut être demandée pour les contenus d'un descripteur ou d'une propriété, d'un événement, d'un signal ou d'une statistique.

Identificateur de terminaison

[,MediaDescriptor]

[,ModemDescriptor](*)

[,MuxDescriptor]

[,EventsDescriptor]

[,SignalsDescriptor]

[,ObservedEventsDescriptor]

[,EventBufferDescriptor]

[,StatisticsDescriptor]

AuditCapabilities(TerminationID,

AuditDescriptor)

(*) Le descripteur ModemDescriptor a été désapprouvé dans la Rec. UIT-T H.248.1 version ~~1~~2 (0~~3~~5/2002).

•••••

Si une réponse remplacée par un caractère générique est demandée, un seul renvoi de commande est généré, et il contient l'union des valeurs de toutes les terminaisons qui correspondent au caractère générique. Cette convention peut réduire le volume de données nécessaires pour auditer un groupe de terminaisons.

Si une propriété, un signal, un événement ou une statistique sont analysés, les propriétés appropriées, les événements concernant les signaux et les statistiques avec les capacités de la terminaison sont renvoyées par la commande AuditCapabilities.

L'interprétation quant aux capacités qui sont requises pour diverses valeurs d'identificateur de contexte et de terminaison est la même que pour la commande AuditValue.

Pour les valeurs de propriété ou de paramètre de type chaîne, caractère ou chaîne d'octets, la passerelle média doit renvoyer une valeur vide. Dans le cas d'un codage alphanumérique, les chaînes et les caractères renvoient une structure quotedString vide, et les chaînes d'octets renvoient NUL (0x00). Ce comportement peut être supplanté par la définition de paquetage.

Le paramètre EventsDescriptor renvoie la liste des événements possibles concernant la terminaison ainsi que la liste de toutes les valeurs possibles pour le paramètre EventsDescriptor. Le paramètre EventBufferDescriptor renvoie les mêmes informations que le paramètre EventsDescriptor, ainsi que la liste de toutes les valeurs possibles pour le paramètre EventsDescriptor. Le paramètre SignalsDescriptor renvoie la liste des signaux dont l'application à la terminaison est possible, ainsi que la liste de toutes les valeurs possibles des paramètres de signaux. Le paramètre StatisticsDescriptor renvoie le nom des statistiques enregistrées pour la terminaison. Le paramètre

ObservedEventsDescriptor renvoie les noms des événements actifs dans la terminaison. Les descripteurs DigitMap et Packages ne sont pas autorisés dans la commande AuditCapability.

•••••

5) Paragraphe 8

Modifier le § 8 comme suit:

8 Transactions

•••••

Les transactions sont présentées sous la forme de demandes de transaction (TransactionRequest). Les réponses correspondantes à une demande de transaction sont reçues dans une seule réponse, éventuellement précédée d'un nombre de messages portant sur les transactions en cours (se reporter au § 8.2.3).

Les transactions garantissent un traitement ordonné des commandes, c'est-à-dire une exécution séquentielle de celles-ci dans le cadre d'une transaction. L'ordre des transactions n'est pas garanti car les transactions peuvent être exécutées dans un ordre quelconque ou simultanément. Toutefois, les réponses à des transactions devraient être traitées avant les demandes de transaction lorsque ces deux types d'éléments figurent dans un message.

Dès la première commande défaillante dans une transaction, le traitement des commandes restant à exécuter dans cette transaction est interrompu. Si une commande contient un identificateur de terminaison remplacé par un caractère générique, cette commande est exécutée avec chacun des identificateurs de terminaison réels qui correspondent à ce caractère. Une réponse dans TransactionReply est incluse pour chaque identificateur de terminaison même si une ou plusieurs instances ont donné lieu à une erreur. Si un identificateur de terminaison correspondant à un caractère générique donne lieu à une erreur lors de son exécution, aucune commande n'est exécutée après celle qui a été remplacée par ce caractère générique.

•••••

8.2.1 Demande de transaction (TransactionRequest)

La demande de transaction est invoquée par l'émetteur. Il y a une seule transaction par invocation de demande. Une demande contient une ou plusieurs actions, dont chacune spécifie son contexte cible~~, ainsi qu'une ou plusieurs commandes par contexte.~~

```
TransactionRequest(TransactionId {
        ContextID {Command … Command},
        . . .
        ContextID  {Command … Command } })
```

Le paramètre TransactionID doit toujours spécifier une valeur en vue d'une corrélation ultérieure avec la réponse de type transaction effectuée ou transaction en cours en provenance du récepteur.

•••••

6) Paragraphe 11.3

Modifier le § 11.3 comme suit:

11.3 Négociation de version de protocole

Une commande ServiceChange provenant d'une passerelle MG qui est enregistrée auprès du contrôleur MGC doit contenir le numéro de version du protocole pris en charge par la passerelle

dans le paramètre ServiceChangeVersion. Quelle que soit la version indiquée dans le paramètre ServiceChangeVersion, le message contenant la commande sera codé comme un message de la version 1. A la réception d'un tel message, si le contrôleur MGC ne prend en charge qu'une version inférieure, le contrôleur MGC doit envoyer une réponse ServiceChangeReply avec la version inférieure et, par la suite, tous les messages entre la passerelle MG et le contrôleur MGC doivent être conformes à la version inférieure du protocole. Si la passerelle MG ne peut pas se conformer et si elle a établi une connexion de transport au contrôleur, il convient que la passerelle ferme cette connexion. Dans tous les cas, il convient qu'elle rejette toutes les demandes subséquentes provenant du contrôleur MGC avec l'erreur 406 (Version non prise en charge).

Si le contrôleur MGC ne prend en charge que des versions de protocole supérieures à celles utilisées par la passerelle MG, il doit rejeter l'association, avec mention de l'erreur 406 (Version non prise en charge).

Si le contrôleur MGC prend en charge la version indiquée par la passerelle MG, il doit se conformer à cette version dans tous les messages ultérieurs. Dans ce cas, il est facultatif que le contrôleur MGC renvoie une version dans la réponse ServiceChangeReply. ~~Si le contrôleur MGC prend en charge une version supérieure à celle de la passerelle MG, il est capable de prendre en charge la version inférieure proposée par la passerelle MG, le contrôleur MGC doit envoyer une réponse ServiceChangeReply avec la version inférieure et, par la suite, tous les messages entre la passerelle MG et le contrôleur MGC doivent être conformes à la version inférieure du protocole. Si le contrôleur MGC ne peut pas se conformer, il doit rejeter l'association, avec l'erreur 406 (Version non prise en charge).~~

La négociation de la version de protocole peut également se produire aux changements de service "handoff" (transfert) et "failover" (reprise sur défaillance).

•••••

7) Paragraphe 12

Modifier le § 12 comme suit:

12.1.5 Statistiques

Statistiques définies par le paquetage, spécifiant:

nom de statistique: élément descriptif seulement

identificateur de statistique: identificateur

identificateur utilisé dans un descripteur de statistique

Description:

type de paramètre: à choisir parmi les suivants:

booléen

chaîne: chaîne en format UTF-8

chaîne d'octets: un certain nombre d'octets. Se reporter aux Annexes A et B.3 en ce qui concerne le codage

entier: entier signé de 4 octets

double: entier signé de 8 octets

caractère: codage de chaque lettre en unicode UTF-8, éventuellement sur plusieurs octets

énumération: liste de valeurs uniques à choisir (se reporter au § 12.3)

sous-liste: liste de plusieurs valeurs tirées d'une liste. Ce type de sous-liste DOIT aussi être spécifié. Le type sera choisi parmi les types indiqués dans le présent paragraphe (à l'exception du type sous-liste). Par exemple, la sous-liste peut être de type sous-liste d'énumération. Le codage des sous-listes est spécifié aux Annexes A et B.3.

Valeurs possibles:

un paquetage doit indiquer l'unité de mesure (par exemple millisecondes, paquets) ici ou avec le type susmentionné, et doit indiquer toute restriction de l'intervalle de validité.

~~unités: unité de mesure, par exemple millisecondes, paquets~~

12.1.6 Procédures

•••••

12.5 Enregistrement de paquetage

Un paquetage peut être enregistré auprès de l'autorité IANA pour des raisons d'interopérabilité. Se reporter au § ~~13~~ 14 pour des considérations relatives à l'autorité IANA.

•••••

8) Annexe B Codage alphanumérique du protocole

•••••

Modifier l'Annexe B comme suit:

B.2 Spécification en formalisme ABNF

La syntaxe du protocole est présentée en formalisme ABNF conformément au commentaire RFC 2234.

NOTE 1 – La présente spécification de syntaxe ne permet pas de faire respecter toutes les restrictions relatives aux inclusions et aux valeurs des éléments. Certaines restrictions supplémentaires sont mentionnées dans des commentaires, tandis que d'autres restrictions apparaissent dans le texte de la présente Recommandation. Ces restrictions supplémentaires font partie du protocole même si leur respect n'est pas assuré par la présente Recommandation.

NOTE 2 – La syntaxe dépend du contexte. Par exemple, "Add" peut être le jeton AddToken ou un nom NAME, en fonction du contexte dans lequel il figure.

Le formalisme ABNF et le codage alphanumérique sont insensibles à la casse. Cela inclut les identificateurs de terminaison, de script de scénario, etc. Le protocole SDP est sensible à la casse en vertu de RFC 2327.

```
; NOTE -- Le formalisme ABNF de la présente section emploie la structure VALUE
; (ou des listes de structures VALUE) pour le codage des diverses valeurs
; d'élément de paquetage (propriétés, paramètres des signaux, etc.). Les types
; de ces valeurs varient et sont spécifiés dans la définition de paquetage
; applicable. Plusieurs de ces types sont décrits au § 12.2.
;
; La spécification du formalisme ABNF pour VALUE admet une forme quotedString ou
; un ensemble de SafeChars. Le codage des valeurs d'élément de paquetage en
; VALUES ABNF est spécifié ci-après. Si le codage d'un type admet des
; caractères autres que SafeChars, la forme quotedString DOIT être employée pour
; toutes les valeurs de ce type, même pour des valeurs spécifiques qui ne
; comportent que des SafeChars.
;
; Chaîne: une chaîne DOIT employer la forme quotedString de VALUE et peut
; contenir tout caractère admis dans cette forme quotedString.
```

```
;
; Entier, double et non signé: les valeurs décimales peuvent être codées au
; moyen des caractères 0 à 9. Les valeurs hexadécimales doivent être précédées
; de '0x' et peuvent utiliser les caractères 0 à 9, a à f et A à F. Le format
; octal n'est pas admis. Les entiers négatifs commençant pas le signe '-'
; DOIVENT être décimaux. La forme SafeChar de VALUE DOIT être employée.
;
; Caractère: un codage UTF-8 d'une seule lettre précédée et suivie de
; guillemets.
;
; Enumération: une énumération DOIT employer la forme SafeChar de VALUE et peut
; contenir tout caractère admis dans cette forme SafeChar.
;
; Booléen: les valeurs booléennes sont codées comme "on" et "off" et ne sont pas
; sensibles à la casse. La forme SafeChar de VALUE DOIT être utilisée.
;
; Types ultérieurs: tout type défini DOIT satisfaire à la spécification ABNF de
; VALUE. En particulier, si le codage d'un type admet des caractères autres que
; SafeChars, la forme quotedString DOIT être employée pour toutes les valeurs de
; ce type, même pour des valeurs spécifiques qui ne comportent que des
; SafeChars.
;
; Il convient de noter qu'il n'y a pas moyen d'employer les guillemets dans une
; valeur.
;
; Il convient aussi de noter que le protocole SDP n'admet pas les blancs en
; début de ligne, tandis que le formalisme ABNF (Megaco) du contrôleur MGC
; autorise les ; blancs avant que le protocole SDP ne débute au niveau du
; descripteur local ou ; distant. Les analyseurs syntaxiques devraient accepter
; les blancs entre le ; LBRKT suivant le jeton local ou distant et le début du
; protocole SDP.

megacoMessage        = LWSP [authenticationHeader SEP ] message

authenticationHeader = AuthToken EQUAL SecurityParmIndex COLON
                       SequenceNum COLON AuthData

SecurityParmIndex    = "0x" 8(HEXDIG)
SequenceNum          = "0x" 8(HEXDIG)
AuthData             = "0x" 24*64(HEXDIG)

message            = MegacopToken SLASH Version SEP mId SEP messageBody
; La version du protocole est définie ici comme étant égale à 2.

messageBody          = ( errorDescriptor / transactionList )

transactionList      = 1*( transactionRequest / transactionReply /
                       transactionPending / transactionResponseAck )
; L'utilisation des acquittements de réponse dépend du transport sous-jacent

transactionPending   = PendingToken EQUAL TransactionID LBRKT RBRKT

transactionResponseAck = ResponseAckToken LBRKT transactionAck
                         *(COMMA transactionAck) RBRKT
transactionAck = TransactionID / (TransactionID "-" TransactionID)

transactionRequest   = TransToken EQUAL TransactionID LBRKT
                       actionRequest *(COMMA actionRequest) RBRKT

actionRequest        = CtxToken EQUAL ContextID LBRKT ((
                       contextRequest [COMMA  commandRequestList])
                       / commandRequestList) RBRKT

contextRequest   = ((contextProperties [COMMA contextAudit])
```

```
                / contextAudit)

contextProperties    = contextProperty *(COMMA contextProperty)

; une fois au plus
; EmergencyOff ne doit être utilisé dans le sens passerelle média vers
; contrôleur de passerelle média que dans le cadre des versions 1 et 2 de la
; Rec. H.248.1.
; EmergencyToken ou EmergencyOffToken, mais pas les deux
contextProperty      = (topologyDescriptor / priority / EmergencyToken/
                        EmergencyOffToken)

contextAudit     = ContextAuditToken LBRKT
                     contextAuditProperties *(COMMA
                     contextAuditProperties) RBRKT

; une fois au plus
contextAuditProperties = ( TopologyToken / EmergencyToken /
                           PriorityToken )

; "O-" indique une commande facultative
; "W-" indique une réponse générique à une commande
commandRequestList= ["O-"] ["W-"] commandRequest *
(COMMA ["O-"] ["W-"]commandRequest)

commandRequest       = ( ammRequest / subtractRequest / auditRequest /
                         notifyRequest / serviceChangeRequest)

transactionReply            = ReplyToken EQUAL TransactionID LBRKT
                                        [ ImmAckRequiredToken COMMA]
                              ( errorDescriptor / actionReplyList ) RBRKT

actionReplyList      = actionReply *(COMMA actionReply )

actionReply          = CtxToken EQUAL ContextID LBRKT
                       ( errorDescriptor / commandReply /
                       (commandReply COMMA errorDescriptor) ) RBRKT

commandReply       = (( contextProperties [COMMA commandReplyList] ) /
                         commandReplyList )


commandReplyList     = commandReplys *(COMMA commandReplys )

commandReplys        = (serviceChangeReply / auditReply / ammsReply /
                         notifyReply )

;Add Move et Modify ont les même paramètres de demande
ammRequest           = (AddToken / MoveToken / ModifyToken ) EQUAL
                       TerminationID [LBRKT ammParameter *(COMMA
                       ammParameter) RBRKT]

;une fois au plus
ammParameter         = (mediaDescriptor / modemDescriptor /
                        muxDescriptor / eventsDescriptor /
                        signalsDescriptor / digitMapDescriptor /
                        eventBufferDescriptor / auditDescriptor)

ammsReply            = (AddToken / MoveToken / ModifyToken /
                        SubtractToken ) EQUAL TerminationID [ LBRKT
                        terminationAudit RBRKT ]

subtractRequest      =  SubtractToken EQUAL TerminationID
                        [ LBRKT auditDescriptor RBRKT]
```

```
auditRequest          =  (AuditValueToken / AuditCapToken ) EQUAL
                         TerminationID LBRKT auditDescriptor RBRKT

auditReply            = (AuditValueToken / AuditCapToken )
                        ( contextTerminationAudit  / auditOther)

auditOther            = EQUAL TerminationID [LBRKT
                        terminationAudit RBRKT]

terminationAudit    = auditReturnParameter *(COMMA auditReturnParameter)

contextTerminationAudit = EQUAL CtxToken ( terminationIDList /
                          LBRKT errorDescriptor RBRKT )

auditReturnParameter = (mediaDescriptor / modemDescriptor /
                         muxDescriptor / eventsDescriptor /
                         signalsDescriptor / digitMapDescriptor /
                       observedEventsDescriptor / eventBufferDescriptor /
                         statisticsDescriptor / packagesDescriptor /
                          errorDescriptor / auditReturnItem)


auditReturnItem       = (MuxToken / ModemToken / MediaToken /
                         DigitMapToken / StatsToken /
                         ObservedEventsToken / PackagesToken)
auditDescriptor       = AuditToken LBRKT [ auditItem
                        *(COMMA auditItem) ] RBRKT

notifyRequest         = NotifyToken EQUAL TerminationID
                        LBRKT ( observedEventsDescriptor
                              [ COMMA errorDescriptor ] ) RBRKT

notifyReply           = NotifyToken EQUAL TerminationID
                        [ LBRKT errorDescriptor RBRKT ]

serviceChangeRequest = ServiceChangeToken EQUAL TerminationID
                        LBRKT serviceChangeDescriptor RBRKT

serviceChangeReply    = ServiceChangeToken EQUAL TerminationID
                        [LBRKT (errorDescriptor /
                        serviceChangeReplyDescriptor) RBRKT]

errorDescriptor    = ErrorToken EQUAL ErrorCode
                     LBRKT [quotedString] RBRKT

ErrorCode             = 1*4(DIGIT) ; could be extended

TransactionID         = UINT32

mId                   = (( domainAddress / domainName )
                        [":" portNumber]) / mtpAddress / deviceName

; Le formalisme ABNF autorise deux ou plusieurs "." consécutifs bien que cela
; n'ait pas de sens dans un mom de domaine.
domainName            = "<" (ALPHA / DIGIT) *63(ALPHA / DIGIT / "-" /
                        ".") ">"
deviceName            = pathNAME

;Les valeurs 0x0, 0xFFFFFFFE et 0xFFFFFFFF sont réservées.
;'-' est utilisé dans le contexte NULL.
ContextID             = (UINT32 / "*" / "-" / "$")

domainAddress         = "[" (IPv4address / IPv6address) "]"
```

```
;RFC 2373 contient la définition de IP6Addresses.
IPv6address          = hexpart [ ":" IPv4address ]
IPv4address          = V4hex DOT V4hex DOT V4hex DOT V4hex
V4hex                = 1*3(DIGIT) ; "0".."255"
; cette production, tout en apparaissant dans RFC 2373, n'est pas référencée
; IPv6prefix           = hexpart SLASH 1*2DIGIT
hexpart             = hexseq "::" [ hexseq ] / "::" [ hexseq ] / hexseq
hexseq               = hex4 *( ":" hex4)
hex4                 = 1*4HEXDIG

portNumber           = UINT16

; Structure d'adressage de mtpAddress:
; 25 - 15      0
;    |  PC    | NI |
;   24 - 14 bits   2 bits
; Note: 14 bits sont destinés à l'usage international.
; Lorsque le code de point est de 16 ou 24 bits, il existe deux options
; nationales.
; Afin d'aligner les octets de mtpAddress, les bits de plus fort poids seront
; codés comme étant des zéros.
; Un octet sera représenté par 2 chiffres hexadécimaux.

mtpAddress           = MTPToken LBRKT 4*8 (HEXDIG) RBRKT

terminationIDList    = LBRKT TerminationID *(COMMA TerminationID) RBRKT

; La longueur totale de pathNAME ne doit pas dépasser 64 caractères.
pathNAME             = ["*"] NAME *("/" / "*"/ ALPHA / DIGIT /"_" / "$" )
                       ["@" pathDomainName ]

; Le formalisme ABNF autorise deux ou plusieurs "." consécutifs bien que cela
; n'ait pas de sens dans un mom de domaine.
pathDomainName       = (ALPHA / DIGIT / "*" )
                       *63(ALPHA / DIGIT / "-" / "*" / ".")

TerminationID        = "ROOT" / pathNAME / "$" / "*"

mediaDescriptor   = MediaToken LBRKT mediaParm *(COMMA mediaParm) RBRKT

; au plus un terminationStateDescriptor
; et soit streamParm(s), soit streamDescriptor(s), mais pas les deux
mediaParm            = (streamParm / streamDescriptor /
                        terminationStateDescriptor)

; une fois au plus par item
streamParm           = ( localDescriptor / remoteDescriptor /
                        localControlDescriptor )

streamDescriptor     = StreamToken EQUAL StreamID LBRKT streamParm
                       *(COMMA streamParm) RBRKT

localControlDescriptor = LocalControlToken LBRKT localParm
                         *(COMMA localParm) RBRKT

; une fois au plus par item sauf pour propertyParm
localParm            = ( streamMode / propertyParm / reservedValueMode
     / reservedGroupMode )

reservedValueMode     = ReservedValueToken EQUAL ( "ON" / "OFF" )
reservedGroupMode     = ReservedGroupToken EQUAL ( "ON" / "OFF" )

streamMode           = ModeToken EQUAL streamModes
```

```
streamModes          = (SendonlyToken / RecvonlyToken / SendrecvToken /
                       InactiveToken / LoopbackToken )

propertyParm         = pkgdName parmValue
parmValue            = (EQUAL alternativeValue/ INEQUAL VALUE)
alternativeValue     = ( VALUE
                    / LSBRKT VALUE *(COMMA VALUE) RSBRKT
                          ; sublist (i.e. A AND B AND ...)
                    / LBRKT VALUE *(COMMA VALUE) RBRKT
                     ; alternatives (i.e. A OR B OR ...)

                    /  LSBRKT VALUE COLON VALUE RSBRKT )
                     ; range


INEQUAL              = LWSP (">" / "<" / "#" ) LWSP; '#' signfie "différent de"
LSBRKT               = LWSP "[" LWSP
RSBRKT               = LWSP "]" LWSP

; Note - L'octet zéro ne figure pas parmi les caractères autorisés dans la
; chaîne d'octets. Puisque la définition en vigueur se limite au protocole
; SDP, et que l'octet zéro serait un caractère illicite dans ce protocole,
; ceci ne pose pas de problème.
localDescriptor      = LocalToken LBRKT octetString RBRKT

remoteDescriptor     = RemoteToken LBRKT octetString RBRKT

eventBufferDescriptor= EventBufferToken [ LBRKT eventSpec
                       *( COMMA eventSpec) RBRKT ]

eventSpec     = pkgdName [ LBRKT eventSpecParameter
                  *(COMMA eventSpecParameter) RBRKT ]
eventSpecParameter   = (eventStream / eventOther)

eventBufferControl     = BufferToken EQUAL ( "OFF" / LockStepToken )

terminationStateDescriptor = TerminationStateToken LBRKT
            terminationStateParm *( COMMA terminationStateParm ) RBRKT

; une fois au plus par item sauf pour propertyParm
terminationStateParm =(propertyParm / serviceStates / eventBufferControl )

serviceStates        = ServiceStatesToken EQUAL ( TestToken /
                       OutOfSvcToken / InSvcToken )

muxDescriptor        = MuxToken EQUAL MuxType  terminationIDList

MuxType              = ( H221Token / H223Token / H226Token / V76Token
                         / extensionParameter / Nx64kToken )

StreamID             = UINT16
pkgdName             = (PackageName SLASH ItemID) ;specific item
                            / (PackageName SLASH "*") ;tous les items dans le
                           paquetage
                            / ("*" SLASH "*") ; tous les items pris en charge
                            / par la passerelle MG
PackageName          = NAME
ItemID               = NAME

eventsDescriptor     = EventsToken [ EQUAL RequestID LBRKT
                       requestedEvent *( COMMA requestedEvent ) RBRKT ]

requestedEvent       = pkgdName [ LBRKT eventParameter
                       *( COMMA eventParameter ) RBRKT ]
```

```
; une fois au plus pour chaque KeepActiveToken , eventDM et eventStream
; au plus un parmi embedWithSig ou embedNoSig mais pas les deux
; KeepActiveToken et embedWithSig ne doivent pas être présents tous les deux
eventParameter       = ( embedWithSig / embedNoSig / KeepActiveToken
                       /eventDM / eventStream / eventOther )

embedWithSig         = EmbedToken LBRKT signalsDescriptor
                       [COMMA embedFirst ] RBRKT
embedNoSig       = EmbedToken LBRKT embedFirst RBRKT

; une fois au plus pour chacun
embedFirst      = EventsToken [ EQUAL RequestID LBRKT
                secondRequestedEvent *(COMMA secondRequestedEvent) RBRKT ]

secondRequestedEvent = pkgdName [ LBRKT secondEventParameter
                       *( COMMA secondEventParameter ) RBRKT ]

; une fois au plus pour chacun parmi embedSig , KeepActiveToken, eventDM ou
; eventStream
; KeepActiveToken et embedSig ne doivent pas être présents tous les deux.
secondEventParameter = ( embedSig / KeepActiveToken / eventDM /
                         eventStream / eventOther )

embedSig  = EmbedToken LBRKT signalsDescriptor RBRKT

eventStream          = StreamToken EQUAL StreamID

eventOther           = eventParameterName parmValue

eventParameterName   = NAME

eventDM              = DigitMapToken EQUAL(( digitMapName ) /
                       (LBRKT digitMapValue RBRKT ))

signalsDescriptor    = SignalsToken [ LBRKT ┼ signalParm
                       *(COMMA signalParm)┼ RBRKT]

signalParm           = signalList / signalRequest

signalRequest        = signalName [ LBRKT sigParameter
                       *(COMMA sigParameter) RBRKT ]

signalList           = SignalListToken EQUAL signalListId LBRKT
                       signalListParm *(COMMA signalListParm) RBRKT

signalListId         = UINT16

; exactement un seul signalType, durée une fois au plus et chaque signal
; parameter
signalListParm       = signalRequest

signalName           = pkgdName
; une fois au plus pour sigStream, une fois au plus pour sigSignalType,
; une fois au plus pour sigDuration, chaque signalParameterName au plus une
; fois.
sigParameter       = sigStream / sigSignalType / sigDuration / sigOther
                          / notifyCompletion / KeepActiveToken
sigStream            = StreamToken EQUAL StreamID
sigOther             = sigParameterName parmValue
sigParameterName     = NAME
sigSignalType        = SignalTypeToken EQUAL signalType
signalType           = (OnOffToken / TimeOutToken / BriefToken)
sigDuration          = DurationToken EQUAL UINT16
```

```
notifyCompletion     = NotifyCompletionToken EQUAL (LBRKT
            notificationReason *(COMMA notificationReason) RBRKT)

notificationReason   = ( TimeOutToken / InterruptByEventToken
                               / InterruptByNewSignalsDescrToken
                               / OtherReasonToken )
observedEventsDescriptor = ObservedEventsToken EQUAL RequestID
                  LBRKT observedEvent *(COMMA observedEvent) RBRKT

; temps par événement, car il peut être mémorisé dans le tampon.
observedEvent        = [ TimeStamp LWSP COLON] LWSP
                       pkgdName [ LBRKT observedEventParameter
                       *(COMMA observedEventParameter) RBRKT ]

; une fois au plus pour eventStream, chaque eventParameterName une fois au plus
observedEventParameter = eventStream / eventOther

; dans le cas où AuditCapReply s'applique à tous les événements, RequestID
; devrait être ALL.
RequestID            = ( UINT32 / "*" )

modemDescriptor      = ModemToken (( EQUAL modemType) /
                       (LSBRKT modemType *(COMMA modemType) RSBRKT))
                       [ LBRKT propertyParm
                        *(COMMA propertyParm) RBRKT ]

; une fois au plus sauf pour extensionParameter
modemType            = (V32bisToken / V22bisToken / V18Token /
                        V22Token / V32Token / V34Token / V90Token /
                        V91Token / SynchISDNToken / extensionParameter)

digitMapDescriptor   = DigitMapToken EQUAL
                       ( ( LBRKT digitMapValue RBRKT )
                / (digitMapName [ LBRKT digitMapValue RBRKT ]) )
digitMapName         = NAME
digitMapValue        = ["T" COLON Timer COMMA] ["S" COLON Timer COMMA]
                       ["L" COLON Timer COMMA] ["Z" COLON Timer COMMA]
                       digitMap
Timer                = 1*2DIGIT
; Les unités sont exprimées en secondes pour les temporisations T, S, et L et en
; centaines de millisecondes pour la temporisation Z. Donc T, S,
; et L s'étendent de 1 à 99 secondes, tandis que les intervalles Z s'étendent de
; 100 ms à 9,9 s.
digitMap = (digitString / LWSP "(" LWSP digitStringList LWSP ")" LWSP)
digitStringList      = digitString *( LWSP "|" LWSP digitString )
digitString          = 1*(digitStringElement)
digitStringElement   = digitPosition [DOT]
digitPosition        = digitMapLetter / digitMapRange
digitMapRange        = ("x" / (LWSP "[" LWSP digitLetter LWSP "]" LWSP))
digitLetter          = *((DIGIT "-" DIGIT ) / digitMapLetter)
digitMapLetter       = DIGIT   ;Basic event symbols
                           / %x41-4B / %x61-6B ; a-k, A-K
                           / "L" / "S" / "T" ;Inter-event timers
                                  ; (long, short, start)
                           / "Z"             ;Long duration modifier

; une fois au plus, et DigitMapToken et PackagesToken ne sont pas autorisés
; dans la commande AuditCapabilities
auditItem            = (~~MuxToken / ModemToken / MediaToken /~~ auditReturnItem/
                        SignalsToken /
                        EventBufferToken /
                        ~~DigitMapToken / StatsToken /~~ EventsToken /
                        ~~ObservedEventsToken / PackagesToken ) /~~
                        indAudterminationAudit)
```

```
indAudterminationAudit   = indAudauditReturnParameter
                            *(COMMA indAudauditReturnParameter)

indAudauditReturnParameter = (indAudmediaDescriptor / /
                                        indAudeventsDescriptor /
                                        indAudsignalsDescriptor /
                                        indAuddigitMapDescriptor /
                                        indAudeventBufferDescriptor /
                                        indAudstatisticsDescriptor /
                                        indAudpackagesDescriptor)


indAudmediaDescriptor   = MediaToken LBRKT indAudmediaParm RBRKT

; une fois au plus par item
; un parmi streamParm ou streamDescriptor mais pas les deux
indAudmediaParm            = (indAudstreamParm / indAudstreamDescriptor /
                                 indAudterminationStateDescriptor)

; une fois au plus
indAudstreamParm           = ( indAudlocalControlDescriptor )
; Le traitement SDP est trop complexe pour extraire des éléments individuels
; en vue d'un audit, donc pas d'audit individuel dans le cas local comme dans
; le cas distant

indAudstreamDescriptor     = StreamToken EQUAL StreamID
                                 LBRKT indAudstreamParm RBRKT

indAudlocalControlDescriptor = LocalControlToken LBRKT indAudlocalParm RBRKT

; une fois au plus par item
indAudlocalParm            = ( ModeToken / pkgdName /
                               ReservedValueToken /
                               ReservedGroupToken )


indAudterminationStateDescriptor = TerminationStateToken LBRKT
                            indAudterminationStateParm RBRKT

; une fois au plus par item
indAudterminationStateParm =(pkgdName / ServiceStatesToken / BufferToken)

indAudeventBufferDescriptor = EventBufferToken LBRKT indAudeventSpec RBRKT

indAudeventSpec         = pkgdName [ LBRKT indAudeventSpecParameter RBRKT ]
indAudeventSpecParameter   = (eventStream / eventParameterName)

indAudeventsDescriptor     = EventsToken EQUAL RequestID LBRKT
                           indAudrequestedEvent RBRKT

indAudrequestedEvent       = pkgdName

indAudsignalsDescriptor    = SignalsToken LBRKT [ indAudsignalParm ] RBRKT

indAudsignalParm           = indAudsignalList / indAudsignalRequest

indAudsignalRequest        = signalName
indAudsignalList           = SignalListToken EQUAL signalListId LBRKT
                                 indAudsignalListParm RBRKT

indAudsignalListParm    = indAudsignalRequest

indAuddigitMapDescriptor   = DigitMapToken EQUAL (digitMapName )
```

```
indAudstatisticsDescriptor = StatsToken LBRKT pkgdName RBRKT

indAudpackagesDescriptor   = PackagesToken LBRKT packagesItem RBRKT


serviceChangeDescriptor = ServicesToken LBRKT serviceChangeParm
                          *(COMMA serviceChangeParm) RBRKT

; chaque paramètre une fois au plus, sauf auditItem
; au plus un parmi serviceChangeAddress ou serviceChangeMgcId mais
; pas les deux
; serviceChangeMethod et serviceChangeReason sont REQUIS
serviceChangeParm    = (serviceChangeMethod / serviceChangeReason /
                        serviceChangeDelay / serviceChangeAddress /
                        serviceChangeProfile / extension / TimeStamp /
                        serviceChangeMgcId / serviceChangeVersion /
                        auditItem)

serviceChangeReplyDescriptor = ServicesToken LBRKT
                        servChgReplyParm *(COMMA servChgReplyParm) RBRKT

; une fois au plus. La version est EXIGEE sur la première réponse ServiceChange.
; au plus un parmi serviceChangeAddress ou serviceChangeMgcId mais
; pas les deux
servChgReplyParm     = (serviceChangeAddress / serviceChangeMgcId /
                        serviceChangeProfile / serviceChangeVersion /
                        TimeStamp)
serviceChangeMethod  = MethodToken EQUAL (FailoverToken /
                        ForcedToken / GracefulToken / RestartToken /
                        DisconnectedToken / HandOffToken /
                        extensionParameter)
; serviceChangeReason comporte un code numérique de raison
; et une description alphanumérique facultative.
; serviceChangeReason DOIT être codé au moyen de la forme quotedString de
; value
; quotedString DOIT contenir un code décimal de raison, suivi
; éventuellement d'un blanc unique et d'une chaîne de description
; alphanumérique.
serviceChangeReason  = ReasonToken  EQUAL VALUE
serviceChangeDelay   = DelayToken   EQUAL UINT32
serviceChangeAddress = ServiceChangeAddressToken EQUAL ( mId /
                        portNumber )
serviceChangeMgcId   = MgcIdToken   EQUAL mId
serviceChangeProfile = ProfileToken EQUAL NAME SLASH Version
serviceChangeVersion = VersionToken EQUAL Version
extension            = extensionParameter parmValue

packagesDescriptor   = PackagesToken LBRKT packagesItem
                        *(COMMA packagesItem) RBRKT

Version              = 1*2(DIGIT)
packagesItem         = NAME "-" UINT16

TimeStamp            = Date "T" Time ; per ISO 8601:1988
; Date = yyyymmdd
Date                 = 8(DIGIT)
; Heure = hhmmssss
Time                 = 8(DIGIT)
statisticsDescriptor = StatsToken LBRKT statisticsParameter
                       *(COMMA statisticsParameter ) RBRKT

; une fois au plus par item
statisticsParameter  = pkgdName [EQUAL VALUE]
```

```
topologyDescriptor   = TopologyToken LBRKT topologyTriple
                       *(COMMA topologyTriple) RBRKT
topologyTriple       = terminationA COMMA
                       terminationB COMMA topologyDirection
                       [COMMA eventStream ]
terminationA         = TerminationID
terminationB         = TerminationID
topologyDirection    = BothwayToken / IsolateToken / OnewayToken

priority             = PriorityToken EQUAL UINT16

extensionParameter   = "X"  ("-" / "+") 1*6(ALPHA / DIGIT)

; octetString est utilisé pour décrire le protocole SDP dans RFC 2327.
; Il convient d'être vigilant lorsque CRLF dans le RFC 2327 est utilisé.
; Par sécurité, utiliser EOL dans ce formalisme ABNF.
; Toutes les fois que le caractère "}" apparaît dans le protocole SDP,
; il constitue une séquence d'échappement avec "\", par exemple "\}".
octetString          = *(nonEscapeChar)
nonEscapeChar        = ( "\}" / %x01-7C / %x7E-FF )
; Note – Les guillemets ne sont pas admis dans la forme quotedString.
quotedString         = DQUOTE *(SafeChar / RestChar/ WSP) DQUOTE

UINT16               = 1*5(DIGIT)  ; %x0-FFFF
UINT32               = 1*10(DIGIT) ; %x0-FFFFFFFF

NAME                 = ALPHA *63(ALPHA / DIGIT / "_" )
VALUE                = quotedString / 1*(SafeChar)
SafeChar             = DIGIT / ALPHA / "+" / "-" / "&" /
                       "!" / "_" / "/" / "'" / "?" / "@" /
                       "^" / "`" / "~" / "*" / "$" / "\" /
                       "(" / ")" / "%" / "|" / "."

EQUAL                = LWSP %x3D LWSP ; "="
COLON                = %x3A           ; ":"
LBRKT                = LWSP %x7B LWSP ; "{"
RBRKT                = LWSP %x7D LWSP ; "}"
COMMA                = LWSP %x2C LWSP ; ","
DOT                  = %x2E           ; "."
SLASH                = %x2F           ; "/"
ALPHA                = %x41-5A / %x61-7A ; A-Z / a-z
DIGIT                = %x30-39         ; 0-9
DQUOTE               = %x22            ; " (Double Quote)
HEXDIG               = ( DIGIT / "A" / "B" / "C" / "D" / "E" / "F" )
SP                   = %x20        ; space
HTAB                 = %x09        ; horizontal tab
CR                   = %x0D        ; Carriage return
LF                   = %x0A        ; linefeed
LWSP                 = *( WSP / COMMENT / EOL )
EOL                  = (CR [LF] / LF )
WSP                  = SP / HTAB ; white space
SEP                  = ( WSP / EOL / COMMENT) LWSP
COMMENT              = ";" *(SafeChar/ RestChar / WSP / %x22) EOL
RestChar             = ";" / "[" / "]" / "{" / "}" / ":" / "," / "#" /
                       "<" / ">" / "="

; Les nouveaux jetons ajoutés à sigParameter doivent prendre le format de SPA*
; * peut avoir une forme quelconque, à savoir SPAM
; Les nouveaux jetons ajoutés à eventParameter doivent prendre le format de EPA*
; * peut avoir une forme quelconque, à savoir EPAD

AddToken                  = ("Add"                  / "A")
AuditToken                = ("Audit"                / "AT")
```

```
AuditCapToken               = ("AuditCapability"       / "AC")
AuditValueToken             = ("AuditValue"            / "AV")
AuthToken                   = ("Authentication"        / "AU")
BothwayToken                = ("Bothway"               / "BW")
BriefToken                  = ("Brief"                 / "BR")
BufferToken                 = ("Buffer"                / "BF")
CtxToken                    = ("Context"               / "C")
ContextAuditToken           = ("ContextAudit"          / "CA")
DigitMapToken               = ("DigitMap"              / "DM")
DisconnectedToken           = ("Disconnected"          / "DC")
DelayToken                  = ("Delay"                 / "DL")
DurationToken               = ("Duration"              / "DR")
EmbedToken                  = ("Embed"                 / "EM")
EmergencyToken              = ("Emergency"             / "EG")
EmergencyOffToken           = ("EmergencyOffToken"     / "EGO")
ErrorToken                  = ("Error"                 / "ER")
EventBufferToken            = ("EventBuffer"           / "EB")
EventsToken                 = ("Events"                / "E")
FailoverToken               = ("Failover"              / "FL")
ForcedToken                 = ("Forced"                / "FO")
GracefulToken               = ("Graceful"              / "GR")
H221Token                   = ("H221" )
H223Token                   = ("H223" )
H226Token                   = ("H226" )
HandOffToken                = ("HandOff"               / "HO")
ImmAckRequiredToken         = ("ImmAckRequired"        / "IA")
InactiveToken               = ("Inactive"              / "IN")
IsolateToken                = ("Isolate"               / "IS")
InSvcToken                  = ("InService"             / "IV")
InterruptByEventToken       = ("IntByEvent"            / "IBE")
InterruptByNewSignalsDescrToken
                            = ("IntBySigDescr"         / "IBS")
KeepActiveToken             = ("KeepActive"            / "KA")
LocalToken                  = ("Local"                 / "L")
LocalControlToken           = ("LocalControl"          / "O")
LockStepToken                  = ("LockStep"           / "SP")
LoopbackToken               = ("Loopback"              / "LB")
MediaToken                  = ("Media"                 / "M")
MegacopToken                = ("MEGACO"                / "!")
MethodToken                 = ("Method"                / "MT")
MgcIdToken                  = ("MgcIdToTry"            / "MG")
ModeToken                   = ("Mode"                  / "MO")
ModifyToken                 = ("Modify"                / "MF")
ModemToken                  = ("Modem"                 / "MD")
MoveToken                   = ("Move"                  / "MV")
MTPToken                    = ("MTP")
MuxToken                    = ("Mux"                   / "MX")
NotifyToken                 = ("Notify"                / "N")
NotifyCompletionToken       = ("NotifyCompletion"/ "NC")
Nx64kToken                  = ("Nx64Kservice"          / "N64")
ObservedEventsToken         = ("ObservedEvents"        / "OE")
OnewayToken                 = ("Oneway"                / "OW")
OnOffToken                  = ("OnOff"                 / "OO")
OtherReasonToken            = ("OtherReason"           / "OR")
OutOfSvcToken               = ("OutOfService"          / "OS")
PackagesToken               = ("Packages"              / "PG")
PendingToken                = ("Pending"               / "PN")
PriorityToken               = ("Priority"              / "PR")
ProfileToken                = ("Profile"               / "PF")
ReasonToken                 = ("Reason"                / "RE")
RecvonlyToken               = ("ReceiveOnly"           / "RC")
ReplyToken                  = ("Reply"                 / "P")
RestartToken                = ("Restart"               / "RS")
RemoteToken                 = ("Remote"                / "R")
```

```
ReservedGroupToken          = ("ReservedGroup"          / "RG")
ReservedValueToken          = ("ReservedValue"          / "RV")
SendonlyToken               = ("SendOnly"               / "SO")
SendrecvToken               = ("SendReceive"            / "SR")
ServicesToken               = ("Services"               / "SV")
ServiceStatesToken          = ("ServiceStates"          / "SI")
ServiceChangeToken          = ("ServiceChange"          / "SC")
ServiceChangeAddressToken   = ("ServiceChangeAddress"   / "AD")
SignalListToken             = ("SignalList"             / "SL")
SignalsToken                = ("Signals"                / "SG")
SignalTypeToken             = ("SignalType"             / "SY")
StatsToken                  = ("Statistics"             / "SA")
StreamToken                 = ("Stream"                 / "ST")
SubtractToken               = ("Subtract"               / "S")
SynchISDNToken              = ("SynchISDN"              / "SN")
TerminationStateToken       = ("TerminationState"       / "TS")
TestToken                   = ("Test"                   / "TE")
TimeOutToken                = ("TimeOut"                / "TO")
TopologyToken               = ("Topology"               / "TP")
TransToken                  = ("Transaction"            / "T")
ResponseAckToken              = ("TransactionResponseAck"/ "K")
V18Token                    = ("V18")
V22Token                    = ("V22")
V22bisToken                 = ("V22b")
V32Token                    = ("V32")
V32bisToken                 = ("V32b")
V34Token                    = ("V34")
V76Token                    = ("V76")
V90Token                    = ("V90")
V91Token                    = ("V91")
VersionToken                = ("Version"                / ——"V")
```

•••••

9) Annexe C Etiquettes des propriétés de flux média

•••••

Modifier l'Annexe C comme suit:

C.1 Attributs généraux des médias

Identificateur de propriété	Etiquette de propriété	Type	Valeur
Gain	100C	~~Entier non signé~~	Inutilisé. Voir le § E.13 en ce qui concerne une propriété de disponibilité gain ~~Gain en dB: 0..65535~~
Ptime	1010	Entier	Temps de mise en paquets Il correspond à la durée en millisecondes du média dans un paquet. Réf.: IETF RFC 2327

C.2 Propriétés Mux

•••••

C.12 H.245

Identificateur de propriété	Etiquette de propriété	Type	Valeur

•••••

CLCack	C006	Octet string	Valeur de la structure H.245 CloseLogicalChannelAck. Réf.: Rec. UIT-T H.245
LCN	C007	Entier	Valeur du numéro de voie locale H.245 (0-65535). Réf.: Rec.UIT-T H.245

Annexe D

Transport sur protocole IP

•••••

10) Annexe E Paquetages de base

•••••

Modifier l'Annexe E comme suit:

E.11 Paquetage de réseau

Identificateur de paquetage: nt (0x000b)

Version: 1

Extension: aucune

Ce paquetage définit les propriétés de terminaisons de réseau indépendantes du type de réseau. Il s'applique, sans s'y limiter, aux réseaux TDM, IP et ATM.

•••••

E.11.4 Statistiques

Durée

Identificateur de statistiques: dur (0x0001)

Description: indique la durée pendant laquelle la terminaison a existé ou n'a pas été dans le contexte NULL~~a été dans le contexte~~.

Type de paramètre: nombre à double précision, en millisecondes

•••••

E.12.4 Statistiques

•••••

Gigue

Identificateur de statistique: jit (0x0007)

Demande la valeur actuelle de la gigue entre arrivées dans un flux RTP tel que défini par le document RFC 1889. La gigue mesure la variation du temps entre arrivées de paquets de données RTP.

Type: double

Valeurs possibles: tout entier de 64 bits

Temps de propagation

Identificateur de statistique: delay (0x0008)

Demande la valeur actuelle du temps de propagation des paquets, exprimée en unités de marqueurs temporels. Même grandeur que la latence moyenne.

Type: double

Valeurs possibles: tout entier de 64 bits

•••••

E.13.1 Propriétés

•••••

Commande de gain

Identificateur de propriété: gain (0x000a)

La commande de gain, ou l'utilisation de l'adaptation du niveau des signaux et de la réduction du niveau de bruit, est utilisée pour adapter le niveau du signal de sortie. Il est cependant nécessaire, par exemple pour les communications par modem, de désactiver cette fonction. Lorsque la valeur est mise à "automatique", la terminaison sert de commande automatique de niveau (ALC, *automatic level control*) dont le niveau cible est fourni par la passerelle média et dont la direction est dirigée vers l'extérieur.

Type de paramètre: entier

Valeurs possibles:

le paramètre de commande de gain spécifie le gain en décibels (valeur positive ou négative), la valeur entière positive maximale 214748647 (0x7fffffff) étant réservée à la représentation de la valeur "automatique". ~~peut être, soit spécifié avec la valeur "automatique" (0xffffffff), soit sous forme de nombre explicite de décibels de gain (toute autre valeur d'entier)~~. La valeur par défaut est fournie dans la passerelle MG.

Défini dans: LocalControlDescriptor

Caractéristiques: lecture/écriture

•••••

11) Appendice I Exemples de flux de communication

•••••

Modifier l'Appendice I comme suit:

I.1.1 Programmation du comportement de repos de terminaisons de ligne analogique passant par une passerelle résidentielle

On trouvera ci-dessous la description des invocations d'interface API issues du contrôleur MGC et des passerelles MG pour faire en sorte que les terminaisons de ce scénario soient programmées au comportement de repos. La passerelle MG d'origine comme son homologue de destination a des terminaisons de ligne analogique au repos qui sont programmées pour rechercher les événements de lancement d'appel (c'est-à-dire de décrochage) au moyen de la commande Modify contenant les paramètres appropriés. Le contexte néant est utilisé pour indiquer que les terminaisons ne sont pas encore associées à un contexte. La terminaison racine est utilisée pour renvoyer à la passerelle MG entière et non pas à une terminaison contenue dans cette passerelle.

Dans cet exemple, la passerelle MG1 possède l'adresse IP 124.124.124.222. La passerelle MG2 possède l'adresse IP 125.125.125.111 et le contrôleur MGC 123.123.123.4. L'accès Megaco par défaut est 55555 pour les trois entités.

1) Une passerelle MG s'enregistre avec un contrôleur MGC au moyen de la commande ServiceChange:

```
MG1 à MGC:

MEGACO/1 [124.124.124.222]
Transaction = 9998 {
    Context = - {
        ServiceChange = ROOT {Services {
            Method=Restart, Version=2,
            ServiceChangeAddress=55555, Profile=ResGW/1}
        }
    }
}
```

2) Le contrôleur MGC envoie la réponse suivante:

```
MGC à MG1:
MEGACO/1 [123.123.123.4]:55555
Reply = 9998 {
   Context = - {ServiceChange = ROOT {
     Services {ServiceChangeAddress=55555, Profile=ResGW/1} } }
}
```

3) Le contrôleur MGC programme une terminaison dans le contexte NULL. L'identificateur de cette terminaison est A4444, l'identificateur de flux est 1, l'identificateur de demande contenu dans le descripteur d'événements est 2222. L'identificateur mId est celui de l'expéditeur de ce message, dans ce cas, il s'agit de l'adresse IP et de l'accès [123.123.123.4]:55555. Le paramètre Mode pour ce flux est réglé à SendReceive (envoyer et recevoir). Le paramètre "al" est le paquetage de supervision de la ligne analogique. On suppose que tant l'option Local que Remote sont assurées.

```
MGC à MG1:
MEGACO/~~1~~2 [123.123.123.4]:55555
Transaction = 9999 {
    Context = - {
        Modify = A4444 {
            Media { Stream = 1 {
                     LocalControl {
                         Mode = SendReceive,
                         tdmc/gain=2,  ; in dB,
                         tdmc/ec=on
                     },
                 }
            },
            Events = 2222 {al/of ~~(~~{strict=state}~~)~~}
        }
```

```
    }
}
```

Le script de numérotation peut avoir été chargé préalablement dans la passerelle MG. Sa fonction sera d'attendre le décrochage, d'activer la tonalité d'invitation et de commencer à acquérir des chiffres par tonalités DTMF. Dans cet exemple cependant, l'on utilise le script de numérotation, qui est mis en place après la détection du décrochage (étape 5) ci-dessous).

Noter que le descripteur d'événements incorporé peut avoir été utilisé pour combiner les étapes 3) et 4) avec les étapes 8) et 9), ce qui élimine les étapes 6) et 7).

4) La passerelle MG1 accepte la commande Modify avec cette réponse:

```
MG1 à MGC:
MEGACO/1—2 [124.124.124.222]:55555
Reply = 9999 {
   Context = - {Modify = A4444}
}
```

5) Un échange similaire se déroule entre la passerelle MG2 et le contrôleur MGC, ce qui produit une terminaison au repos appelée A5555.

I.1.2 Acquisition des chiffres de l'expéditeur et de la terminaison d'origine

La procédure suivante est fondée sur les conditions indiquées ci-dessus. Elle décrit les transactions issues du contrôleur MGC et de la passerelle média d'origine (MG1, *originating media gateway*) afin d'acquérir la terminaison d'origine (A4444) en passant par les étapes d'acquisition de chiffres nécessaires pour ouvrir une connexion vers la passerelle média de destination (MG2, *terminating media gateway*).

6) La passerelle MG1 détecte un événement de décrochage en provenance de l'utilisateur 1 et le signale au contrôleur MGC au moyen de la commande Notify.

```
MG1 à MGC:
MEGACO/1—2 [124.124.124.222]:55555
Transaction = 10000 {
   Context = - {
       Notify = A4444 {ObservedEvents =2222 {
         19990729T22000000:al/of(init=false)}}
   }
}
```

7) Et la commande Notify est acquittée.

```
MGC à MG1:
MEGACO/1—2 [123.123.123.4]:55555
Reply = 10000 {
    Context = - {Notify = A4444}
}
```

8) Le contrôleur MGC modifie la terminaison afin de jouer la tonalité de numérotation, de rechercher des chiffres conformément à Dialplan0 et de chercher maintenant un événement de raccrochage.

```
MGC à MG1:
MEGACO/1—2 [123.123.123.4]:55555
Transaction = 10001 {
    Context = - {
        Modify = A4444 {
            Events = 2223 {
                al/on(strict=state), dd/ce {DigitMap=Dialplan0}
            },
            Signals {cg/dt},
            DigitMap= Dialplan0{
```

```
(0| 00|[1-7]xxx|8xxxxxxx|Fxxxxxxx|Exx|91xxxxxxxxx|9011x.)}
        }
    }
}
```

9) Et la commande Modify est acquittée.

```
MG1 to MGC:
MEGACO/1 2 [124.124.124.222]:55555
Reply = 10001 {
    Context = - {Modify = A4444}
}
```

10) Ensuite, les chiffres sont accumulés par la passerelle MG1 au fur et à mesure qu'ils sont composés par l'utilisateur 1. La tonalité de numération (Dialtone) est arrêtée à la détection du premier chiffre. Lorsqu'une correspondance appropriée est réalisée pour les chiffres recueillis avec le plan de numérotation couramment programmé (Dialplan) pour A4444, une autre commande Notify est envoyée au contrôleur de passerelle média.

```
MG1 à MGC:
MEGACO/1 2 [124.124.124.222]:55555
Transaction = 10002 {
   Context = - {
       Notify = A4444 {ObservedEvents =2223 {
         19990729T22010001:dd/ce{ds="916135551212",Meth=UM}}}
   }
}
```

11) Et la commande Notify est acquittée.

```
MGC à MG1:
MEGACO/1 2 [123.123.123.4]:55555
Reply = 10002 {
    Context = - {Notify = A4444}
}
```

12) Le contrôleur analyse ensuite les chiffres et détermine qu'une connexion doit être établie de la passerelle MG1 à la passerelle MG2. Aussi bien la terminaison TDM A4444 qu'une terminaison RTP sont ajoutées à un nouveau contexte dans la passerelle MG1. Le mode est réception seulement car les valeurs du descripteur Remote ne sont pas encore spécifiées. Les codecs préférés sont énumérés dans l'ordre de préférence du contrôleur MGC.

```
MGC à MG1:
MEGACO/1 2 [123.123.123.4]:55555
Transaction = 10003 {
    Context = $ {
       Add = A4444,
       Add = $ {
           Media {
             Stream = 1 {
                  LocalControl {
                      Mode = ReceiveOnly,

                      nt/jit=40 ; in ms
                  },
                  Local {
v=0
c=IN IP4 $
m=audio $ RTP/AVP 4
a=ptime:30
v=0
c=IN IP4 $
m=audio $ RTP/AVP 0
                  }
```

```
            }
          }
        }
    }
}
```

NOTE – Le contrôleur MGC indique ses valeurs de paramètre préférées comme une série de blocs SDP contenue dans le descripteur Local. Celle-ci insère le descripteur Local dans la réponse.

13) La passerelle MG1 acquitte la nouvelle terminaison et insère l'adresse IP locale et l'accès UDP. Elle effectue également, dans le descripteur Local, un choix de codec sur la base des préférences du contrôleur MGC. La passerelle MG1 règle l'accès RTP à la valeur 2222.

```
MEGACO/~~1~~2 [124.124.124.222]:55555
Reply = 10003 {
   Context = 2000 {
      Add = A4444,
      Add=A4445{
         Media {
             Stream = 1 {
                 Local {
v=0
o=- 2890844526 2890842807 IN IP4 124.124.124.222
s=-
t= 0 0
c=IN IP4 124.124.124.222
m=audio 2222 RTP/AVP 4
a=ptime:30
a=recvonly
                 } ; le profil RTP pour G.723.1 est 4
             }
         }
      }
   }
}
```

14) Le contrôleur MGC associera désormais la terminaison A5555 avec un nouveau contexte dans la passerelle MG2 et établira vers l'utilisateur d'origine (utilisateur 1) une connexion d'émission-réception avec identification dans le flux RTP (c'est-à-dire que la terminaison A5556 sera assignée). Le contrôleur MGC règle également la sonnerie sur A5555.

```
MGC à MG2:
MEGACO/~~1~~2 [123.123.123.4]:55555
Transaction = 50003 {
    Context = $ {
       Add = A5555  { Media {
            Stream = 1 {
                 LocalControl {Mode = SendReceive} }},
         Events=1234{al/of(strict=state)},
            Signals {al/ri}
            },
       Add  = $ {Media {
            Stream = 1 {
                 LocalControl {
                    Mode = SendReceive,
                    nt/jit=40 ; in ms
                 },
                 Local {
v=0
c=IN IP4 $
m=audio $ RTP/AVP 4
a=ptime:30
                 },
```

```
                  Remote {
v=0
c=IN IP4 124.124.124.222
m=audio 2222 RTP/AVP 4
a=ptime:30
                  } ; le profil RTP pour G.723.1 est 4
               }
            }
         }
      }
}
```

15) Cette connexion est acquittée. Le numéro d'accès au flux est différent du numéro d'accès de commande. Dans ce cas, ce numéro est 1111 (dans le protocole SDP).

```
MG2 à MGC:
MEGACO/~~1~~2 [125.125.125.111]:55555
Reply = 50003 {
   Context = 5000 {
     Add = A5555,
      Add = A5556{
         Media {
            Stream = 1 {
                Local {
v=0

o=- 7736844526 7736842807 IN IP4 125.125.125.111
s=-
t= 0 0
c=IN IP4 125.125.125.111
m=audio 1111 RTP/AVP 4
}
            } ; le profil RTP pour G.723.1 est 4
         }
      }
   }
}
```

16) L'adresse IP et l'accès UDP ci-dessus définis doivent maintenant être indiqués à la passerelle MG1.

```
MGC à MG1:
MEGACO/~~1~~2 [123.123.123.4]:55555
Transaction = 10005 {
  Context = 2000 {
    Modify = A4444 {
      Signals {cg/rt}
    },
    Modify = A4445 {
       Media {
            Stream = 1 {
                Remote {
v=0

o=- 7736844526 7736842807 IN IP4 125.125.125.111
s=-
t= 0 0
c=IN IP4 125.125.125.111
m=audio 1111 RTP/AVP 4
                }
            } ; le profil RTP pour G.723.1 est 4
         }
    }
  }
```

```
}

MG1 à MGC:
MEGACO/1—2 [124.124.124.222]:55555
Reply = 10005 {
   Context = 2000 {Modify = A4444, Modify = A4445}
}
```

17) Les deux passerelles sont maintenant connectées et l'utilisateur 1 entend le retour d'appel. La passerelle MG2 attend maintenant que l'utilisateur 2 soulève le combiné pour que la communication soit établie dans les deux sens.

```
De MG2 à MGC:

MEGACO/1—2 [125.125.125.111]:55555
Transaction = 50005 {
   Context = 5000 {
       Notify = A5555 {ObservedEvents =1234 {
         19990729T22020002:al/of(init=false)}}
   }
}

De MGC à MG2:

MEGACO/1—2 [123.123.123.4]:55555
Reply = 50005 {
    Context = - {Notify = A5555}
}

De MGC à MG2:

MEGACO/1—2 [123.123.123.4]:55555
Transaction = 50006 {
   Context = 5000 {
      Modify = A5555 {
         Events = 1235 {al/on(strict=state)},
          Signals { } ; pour couper la sonnerie
      }
   }
}

De MG2 à MGC:

MEGACO/1—2 [125.125.125.111]:55555
Reply = 50006 {
 Context = 5000 {Modify = A4445}
}
```

18) Dans la passerelle MG1, le mode est commuté sur émission-réception et le retour d'appel est arrêté.

```
MGC à MG1:
MEGACO/1—2 [123.123.123.4]:55555
Transaction = 10006 {
   Context = 2000 {
      Modify = A4445 {
         Media {
            Stream = 1 {
               LocalControl {
                  Mode=SendReceive
               }
            }
         }
      },
```

```
      Modify = A4444 {
         Signals { }
      }
   }
}

De MG1 à MGC:
MEGACO/1 2 [124.124.124.222]:55555
Reply = 10006 {
   Context = 2000 {Modify = A4445, Modify = A4444}}
```

19) Le contrôleur MGC décide d'envoyer la commande Audit à la terminaison RTP de la passerelle MG2.

```
MEGACO/1 2 [123.123.123.4]:55555
Transaction = 50007 {
   Context = - {AuditValue = A5556{
      Audit{Media, DigitMap, Events, Signals, Packages, Statistics }}
   }
}
```

20) La passerelle MG2 répond.

```
MEGACO/1 2 [125.125.125.111]:55555
Reply = 50007 {
   Context = - {
AuditValue = A5556 {
          Media {
             TerminationState { ServiceStates = InService,
                 Buffer = OFF },
             Stream = 1 {
                 LocalControl { Mode = SendReceive,
                    nt/jit=40 },
                 Local {
v=0

o=- 7736844526 7736842807 IN IP4 125.125.125.111
s=-
t= 0 0
c=IN IP4 125.125.125.111
m=audio 1111 RTP/AVP  4
a=ptime:30
                },
                 Remote {
v=0

o=- 2890844526 2890842807 IN IP4 124.124.124.222
s=-
t= 0 0
c=IN IP4 124.124.124.222
m=audio 2222 RTP/AVP  4
a=ptime:30
                 } } },
          Events,
          Signals,
          DigitMap,
          Packages {nt-1, rtp-1},
          Statistics { rtp/ps=1200,  ; paquets envoyés
                       nt/os=62300, ; octets envoyés
                       rtp/pr=700, ; paquets reçus
                       nt/or=45100, ; octets reçus
                       rtp/pl=0.2,  ; % perte de paquets
                       rtp/jit=20,
```

```
                        rtp/delay=40 } ; latence moyenne
        }
    }
}
```

21) Lorsque le contrôleur MGC reçoit un signal de raccrochage en provenance d'une des passerelles MG, il libère la communication. Dans cet exemple, l'utilisateur situé du côté de la passerelle MG2 raccroche le premier.

```
De MG2 à MGC:

MEGACO/1 2 [125.125.125.111]:55555
Transaction = 50008 {
   Context = 5000 {
       Notify = A5555 {ObservedEvents =1235 {
          19990729T24020002:al/on(init=false)}
       }
   }
}

De MGC à MG2:

MEGACO/1 2 [123.123.123.4]:55555
Reply = 50008 {
    Context = - {Notify = A5555}
}
```

22) Le contrôleur MGC envoie ensuite aux deux passerelles MG une commande Subtract afin de libérer la communication. Seules les commandes de soustraction envoyées à la passerelle MG2 sont représentées ici. Chaque terminaison possède son propre paquetage de statistiques recueillies. Un contrôleur MGC peut ne pas avoir besoin de demander que les deux commandes lui soient renvoyées. La terminaison A5555 est physique, tandis que la terminaison A5556 est en protocole RTP.

```
De MGC à MG2:

MEGACO/1 2 [123.123.123.4]:55555
Transaction = 50009 {
   Context = 5000 {
      Subtract = A5555 {Audit{Statistics}},
      Subtract = A5556 {Audit{Statistics}}
   }
}

De MG2 à MGC:

MEGACO/1 2 [125.125.125.111]:55555
Reply = 50009 {
   Context = 5000 {
     Subtract = A5555 {
          Statistics {
             nt/os=45123, ; Octets envoyés
             nt/or=45123, ; Octets reçus
             nt/dur=40000 ; en millisecondes
             }
       },
       Subtract = A5556 {
          Statistics {
             rtp/ps=1245, _; paquets envoyés
             nt/os=62345, _; octets envoyés
             rtp/pr=780, __; packets reçus
             nt/or=45123, _; octets reçus
```

```
                rtp/pl=10, ___; –% perte de paquets
                rtp/jit=27,
                rtp/delay=48, ; latence moyenne
                nt/dur=38000  ; en millisecondes
            }
        }
    }
}
```

23) Le contrôleur MGC configure ensuite les deux passerelles MG1 et MG2 de façon qu'elles soient prêtes à détecter le prochain événement de décrochage. Se reporter à l'étape 1). Noter que cet état pourrait être défini comme étant la valeur par défaut d'une terminaison dans le contexte néant. Dans ce cas, aucun message n'a besoin d'être envoyé du contrôleur MGC à la passerelle MG. Une fois qu'une terminaison revient au contexte néant, elle reprend ses valeurs par défaut assignées.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	**Systèmes audiovisuels et multimédias**
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, circuits téléphoniques, télégraphie, télécopie et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de nouvelle génération
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse
Genève, 2004