UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Q.754

(06/97)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE Q: COMMUTATION ET SIGNALISATION

Spécifications du système de signalisation n° 7 – Gestion du système de signalisation n° 7

Définitions des éléments de service d'application pour la gestion du système de signalisation n° 7

Recommandation UIT-T Q.754

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE Q

COMMUTATION ET SIGNALISATION

SIGNALISATION DANS LE SERVICE MANUEL INTERNATIONAL	Q.1–Q.3
EXPLOITATION INTERNATIONALE AUTOMATIQUE ET SEMI-AUTOMATIQUE	Q.4–Q.59
FONCTIONS ET FLUX D'INFORMATION DES SERVICES DU RNIS	Q.60–Q.99
CLAUSES APPLICABLES AUX SYSTÈMES NORMALISÉS DE L'UIT-T	Q.100–Q.119
SPÉCIFICATIONS DES SYSTÈMES DE SIGNALISATION N° 4 ET N° 5	Q.120–Q.249
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION N° 6	Q.250–Q.309
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION R1	Q.310–Q.399
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION R2	Q.400–Q.499
COMMUTATEURS NUMÉRIQUES	Q.500–Q.599
INTERFONCTIONNEMENT DES SYSTÈMES DE SIGNALISATION	Q.600–Q.699
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION N° 7	Q.700–Q.849
Généralités	Q.700
Sous-système transport de messages	Q.701–Q.709
Sous-système commande des connexions sémaphores	Q.711–Q.719
Sous-système utilisateur téléphonie	Q.720–Q.729
Services complémentaires du RNIS	Q.730–Q.739
Sous-système utilisateur données	Q.740–Q.749
Gestion du système de signalisation n° 7	**Q.750–Q.759**
Sous-système utilisateur du RNIS	Q.760–Q.769
Sous-système application de gestion des transactions	Q.770–Q.779
Spécification des tests	Q.780–Q.799
Interface Q3	Q.800–Q.849
SYSTÈME DE SIGNALISATION D'ABONNÉ NUMÉRIQUE N° 1	Q.850–Q.999
RÉSEAUX MOBILES TERRESTRES PUBLICS	Q.1000–Q.1099
INTERFONCTIONNEMENT AVEC LES SYSTÈMES MOBILES À SATELLITES	Q.1100–Q.1199
RÉSEAU INTELLIGENT	Q.1200–Q.1999
RNIS À LARGE BANDE	Q.2000–Q.2999

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

RECOMMANDATION UIT-T Q.754

DEFINITIONS DES ELEMENTS DE SERVICE D'APPLICATION POUR LA GESTION DU SYSTEME DE SIGNALISATION N° 7

Résumé

La présente Recommandation définit l'élément de service d'application (ASE) utilisé par les fonctions de gestion MRVT, SRVT et CVT décrites dans la Recommandation Q.753. L'élément ASE définit l'information de gestion utilisée par ces fonctions dans les messages traversant le réseau du système de signalisation n° 7.

L'élément ASE est connecté au gestionnaire de transactions (TC) pour fournir les services utilisés par l'utilisateur OMASE (défini dans la Recommandation Q.753), pour permettre des communications entre les nœuds du réseau sémaphore à l'aide des fonctions MRVT, SRVT ou CVT.

Les principales révisions par rapport à la version 1993 de la présente Recommandation sont les suivantes:

a) révision des règles de compatibilité pour permettre le transport transparent de paramètres non reconnus;

b) définition d'un nouveau paramètre de message MRVT et SRVT pour spécifier qu'une information est nécessaire dans tout message MRVR ou SRVR;

c) définition d'un nouveau message MRVR et SRVR à utiliser si d'autres informations que celles qui sont spécifiées pour les messages MRVR et SRVR de la version 1993 sont nécessaires;

d) définition de nouveaux paramètres de message MRVT et MRVR pour indiquer les priorités en matière d'acheminement;

e) définition de nouveaux paramètres MRVA, MRVR, SRVA et SRVR pour permettre le renvoi des paramètres non reconnus dans les messages MRVT ou SRVT;

f) définition d'un nouveau paramètre MRVT demandant aux nœuds s'ils disposent d'une voie d'acheminement à destination de l'initiateur de test passant par le nœud en provenance duquel ils ont reçu le message MRVT (cela permet de vérifier la symétrie des voies d'acheminement).

Source

La Recommandation UIT-T Q.754, révisée par la Commission d'études 11 de l'UIT-T (1997-2000), a été approuvée le 5 juin 1997 selon la procédure définie dans la Résolution n° 1 de la CMNT.

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.

La Conférence mondiale de normalisation des télécommunications (CMNT), qui se réunit tous les quatre ans, détermine les thèmes d'études 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 n° 1 de la CMNT.

Dans certains secteurs de la technologie 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.

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/n'avait pas 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

Recommandation Q.754

DEFINITIONS DES ELEMENTS DE SERVICE D'APPLICATION POUR LA GESTION DU SYSTEME DE SIGNALISATION N° 7

(révisée en 1997)

1 Introduction

Il convient de noter qu'en cas de conflit entre les Recommandations Q.753 et Q.754, cette dernière aura priorité.

La présente Recommandation définit l'élément de service d'application (ASE) du sous-système d'exploitation, de gestion et d'administration (OMAP), appelé OMASE. L'élément OMASE fournit les services invoqués à travers la frontière entre l'élément OMASE et l'utilisateur de cet élément au moyen des primitives OM-EVENT-REPORT et OM-CONFIRMED-ACTION. Se référer à la Recommandation Q.753 pour un diagramme et le mappage entre les services demandés au sein de l'utilisateur OMASE et ceux de l'élément OMASE.

Les services OMASE sont dérivés de ceux définis dans le protocole CMIP[1].

Les primitives OMASE sont définies dans le paragraphe 6, leur syntaxe formelle définie dans la Figure 3 utilise les capacités du gestionnaire de transaction (TC, *transaction capabilities*) OPERATION et ERROR. Le paragraphe 6 décrit également l'interaction entre l'élément OMASE et le gestionnaire de transaction.

L'élément OMASE fournit des opérations permettant l'administration du réseau à partir du processus de gestion OMASE et de l'utilisateur OMASE, en vue de réaliser les tests de vérification d'acheminement des sous-systèmes MTP et SCCP (MRVT et SRVT) et les tests de validation de circuit (CVT, *circuit validation tests*). La présente Recommandation contient la définition ASE pour MRVT, SRVT et CVT.

Les tests SRVT auxquels il est fait référence dans ce document sont destinés aux tests spécifiques du 3.2.2/Q.753.

Les arguments utilisés pour des primitives émises à travers les frontières entre le processus de gestion du sous-système OMAP vers l'utilisateur OMASE, pour les primitives à travers la frontière entre l'utilisateur OMASE et le sous-système OMAP ainsi qu'entre l'élément OMASE et le gestionnaire de transaction contiennent la même information s'ils portent le même nom. Ces arguments sont définis dans la présente Recommandation.

Les messages entre points sémaphores sont codés au moyen des règles de codage de base (BER, *basic encoding rules*) de la notation ASN.1, tandis que les paramètres de type chaîne d'octets sont codés sous la forme d'éléments de primitive (et non sous celle de structures syntaxiques).

---

[1] Le protocole CMIP est défini dans l'ISO/CEI 9596 et dans la Recommandation X.711.

2 Sous-système MTP

2.1 Test de vérification d'acheminement du sous-système MTP (MRVT)[2]

Le test MRVT lancé à l'origine du test génère une primitive OM-CONFIRMED-ACTION utilisée par l'utilisateur OMASE vers l'élément OMASE, comportant comme paramètre la commande "test d'itinéraire" (*testRoute*). La primitive OM-EVENT-REPORT, contenant comme paramètre l'événement "trace d'itinéraire" (*routeTrace*) ou "nouvelle trace d'itinéraire" (*routeTraceNew*), est invoquée pour l'élément OMASE par l'initiateur du test si une trace des itinéraires est demandée, ou en cas de faute.

L'action "test d'itinéraire" est spécifiée au moyen de la macro CNF-ACTION définie dans la Figure 3, l'événement "trace d'itinéraire" est spécifié au moyen de la macro EVENT définie dans la Figure 3. La Figure 3 constitue la définition du module OMASE et définit tous les paramètres utilisés dans ce protocole.

Dans le cas du test MRVT, la classe d'objets indique les tables d'acheminement du sous-système MTP et l'instance de l'objet contient le code de point de la destination du test. L'action "test d'itinéraire" utilise le message BEGIN (du test MRVT) avec un résultat (MRVA) renvoyé dans un message END. L'événement "trace d'itinéraire" (du test MRVR) utilise un message BEGIN avec une fin prédéterminée.

2.1.1 Action de test d'itinéraire (*testRoute*)

L'action "test d'itinéraire" est invoquée pour lancer un test de vérification d'acheminement du sous-système MTP. Au niveau du nœud d'origine, cette invocation est demandée par l'Administration par l'intermédiaire de l'utilisateur MIS ou d'une interface locale à travers le processus de gestion du sous-système OMAP et l'utilisateur OMASE. Dans les nœuds suivants, l'action est demandée d'une manière implicite par la réception d'une invocation de l'action "test d'itinéraire". Une réponse positive indique la réussite du test au point où il a été invoqué, et d'une manière implicite, à tous les points suivants où le test à été invoqué. Une indication de faute est renvoyée pour indiquer que le test a échoué au niveau de ce nœud ou d'un nœud suivant.

testRoute CNF-ACTION	*Temporisation* = T1	*Classe* = 1	*Code* = 00000001

Voir la Figure 3.

2.1.1.1 Arguments de l'action de test d'itinéraire

2.1.1.1.1 Point sémaphore initiateur (*initiatingSP*)

Le paramètre "point sémaphore initiateur" identifie l'initiateur du test. Il est du type "code de point" (*pointCode*), défini comme une chaîne d'octets.

Paramètre	*Code*
initiatingSP	10000000
Contenu	
le bit 0 contient le premier bit du code de point. le bit 1 contient le deuxième bit du code de point, etc.	

[2] Voir les Recommandations X.680 à X.683 et X.690 pour la description de la notation formelle (ainsi que les Recommandations X.208 et X.209).

2.1.1.1.2 Trace demandée (*traceRequested*)

Le paramètre "trace demandée" indique qu'une trace de tous les itinéraires utilisés pour atteindre la destination doit faire l'objet d'un compte rendu à l'initiateur (l'événement "trace d'itinéraire" est décrit au 2.1.2). Il est du type BOOLEAN.

Paramètre	*Code*
traceRequested	10000001
Contenu	*Signification*
"Vrai" (= 1)	une trace a été demandée, renvoi de l'information de trace en cas de succès et de faute.
"Faux" (= 0)	pas de trace demandée, renvoi de l'information de trace uniquement en cas de faute.

2.1.1.1.3 Seuil (*threshold*)

L'initiateur définit un seuil maximal pour le nombre de points sémaphores (SP, *signalling points*) qui peuvent être traversés au cours du test (y compris l'initiateur s'il est un point de transfert sémaphore). Ceci facilite la détection d'itinéraires de longueur anormale. Ce seuil indique un nombre entier de points sémaphores et son type est INTEGER.

Paramètre	*Code*
threshold	10000010

2.1.1.1.4 Codes de points traversés (*pointCodesTraversed*)

Lorsqu'un point sémaphore est traversé, il ajoute son code de point à la liste des codes de points traversés. Ceci facilite la détection de boucles et constitue également une information utile en cas de défaillance ou lorsqu'une trace d'itinéraire est demandée. Il s'agit d'une liste de codes de point du type "liste de codes de point".

Paramètre	*Code*
pointCodesTraversed	10100011
Contenu	
suite de codes de points, étiquetée "code de point" dont le contenu indique le code de point exact.	

2.1.1.1.5 Liste de priorités d'itinéraire (*routePriorityList*)

Si le paramètre "demande d'information" est présent et en fait la demande, tout point sémaphore traversé ajoute la priorité de l'itinéraire vers le prochain point sémaphore dans la liste de priorités d'itinéraire.

Paramètre	*Code*
routePriorityList	10101100
Contenu	
suite de priorités, étiquetée "Priorité" dont le contenu indique inconnu, premier choix, second choix, etc.	

2.1.1.1.6 Demande d'information (*infoRequest*)

Ce paramètre facultatif, pouvant être inséré uniquement par le point sémaphore initiateur du test, indique que l'initiateur peut reconnaître des messages MRVR créés par le type d'événement "nouvelle trace d'itinéraire". Le paramètre "demande d'information" indique quelle est l'information demandée dans le cas où un message MRVR est envoyé à l'initiateur. Il peut également indiquer quels sont les paramètres qui doivent être mis à jour lorsque les messages MRVT traversent le réseau. Les valeurs courantes peuvent être "code de point" (bit 0 = 1) et/ou "liste de codes de point" (bit 1) et/ou "liste de priorités d'itinéraire" (bit 2).

Paramètre	*Code*
infoRequest	10001101
Contenu	
chaîne de bits contenant une ou plusieurs des valeurs indiquées	

2.1.1.1.7 Retour de paramètres inconnus (*returnUnknownParams*)

Ce paramètre facultatif peut uniquement être inséré par le point sémaphore initiateur du test (et seulement si le paramètre "demande d'information" est également présent). Il indique quels sont les paramètres MRVT qu'un nœud successeur devra renvoyer, si ce successeur ne reconnaît pas les paramètres en question. Les paramètres MRVT inconnus doivent être recopiés dans le (nouveau) message MRVR (paramètre "nouvelle trace d'itinéraire") si le nœud successeur a l'occasion de renvoyer un tel message MRVR, ou dans un paramètre "copie de données" (*copyData*) d'un message MRVA s'il ne connaît pas l'initiateur. Le bit 0 dans le paramètre "retour de paramètres inconnus" indique un paramètre MRVT avec une valeur d'étiquette égale à 15, le bit 1 un paramètre MRVT avec une valeur d'étiquette égale à 16, et ainsi de suite.

Paramètre	*Code*
returnUnknownParams	10001110
Contenu	
chaîne de bits contenant une ou plusieurs valeurs indiquées	

2.1.1.1.8 Vérification d'itinéraire direct (*directRouteCheck*)

Ce paramètre facultatif indique, s'il est positionné sur "Vrai", que les nœuds suivants ont l'obligation de vérifier s'ils disposent, vers l'initiateur du test, d'un itinéraire direct passant par le point sémaphore duquel ils ont reçu le message MRVT instigateur.

Paramètre	*Code*
directRouteCheck	10001111
Contenu	
Booléen, "Vrai" indique que la vérification est demandée	

2.1.1.2 Résultats de l'action

Une indication de retour positive ne contient pas de paramètre.

2.1.1.3 Erreurs de l'action

Les erreurs spécifiques sont des erreurs possibles pouvant survenir pendant un test et qui sont propres à ce test. Ces erreurs spécifiques existent en plus des erreurs déjà identifiées dans le service OM-CONFIRMED-ACTION (*action OM confirmée*) et apparaissent comme paramètres de l'erreur de défaillance de traitement.

2.1.1.3.1 Défaillance (*failure*)

L'erreur spécifique "défaillance" indique une condition de défaillance totale lorsque aucun itinéraire ne fonctionne correctement. Elle sera utilisée le plus souvent comme indication de défaillance à partir du point qui détecte l'erreur et n'invoque pas d'action "test d'itinéraire" ultérieure. L'erreur spécifique "défaillance" véhicule un paramètre permettant d'indiquer la condition d'erreur qui cause la défaillance. Ce paramètre "type de défaillance" (*failureType*) est représenté par une chaîne binaire. Le deuxième paramètre à utiliser conjointement indique l'erreur "point sémaphore initiateur non connu" (*unknownInitiatingSP*). Le paramètre "trace émise" (*traceSent*) indique si un événement "trace d'itinéraire" a été invoqué pour rendre compte d'une information d'itinéraire. Il est nécessaire d'indiquer cette information pour cette erreur, parce que le nœud qui détecte l'erreur ne peut émettre d'événement "trace d'itinéraire" et c'est, en conséquence, le nœud précédent qui doit le faire. Le paramètre "trace émise" est du type BOOLEAN. Ce troisième paramètre est facultatif, il est présent si le paramètre "type de défaillance" est égal à "point sémaphore initiateur non connu", si "trace émise" est "Faux" et si le message MRVT de demande contenait un paramètre "demande d'information" (*requestInfo*) ou un paramètre "retour de paramètres inconnus" (ou les deux).

Erreur spécifique	*Code*
failure	00000001

Paramètre	*Code*
failureType	10000000

Paramètre	*Code*
traceSent	10000001
Contenu	*signification*
"Vrai"	l'information de trace a été émise
"Faux"	l'information de trace n'a pas été émise

Paramètre	*Code*
copyData	10000100
Contenu	
chaîne d'octets contenant les paramètres demandés par le message MRVT de demande	

2.1.1.3.2 Réussite partielle (*partialSuccess*)

Cette indication est fournie lorsque au moins une invocation de l'action "test d'itinéraire" a échoué et qu'au moins une autre a réussi, au moins en partie. Dans ce cas, tout type d'erreur qui s'est manifestée

sera noté et émis dans la réponse finale. Le format et le contenu du paramètre "réussite partielle" sont les mêmes que ceux du paramètre "défaillance".

Erreur spécifique	*Code*
partialSuccess	00000010

2.1.2 Evénement de trace d'itinéraire (*routeTrace*)

L'événement "trace d'itinéraire" rend compte d'une information de trace. L'information de trace se constitue de zéro, un ou plusieurs codes de point, tels que le code de point détectant une erreur ou la liste totale des codes de point traversés par un itinéraire. Cet événement est invoqué soit par le nœud initiateur (indiqué par "trace demandée", voir 2.1.1.1.2) ou par une défaillance survenant en tout point situé sur l'itinéraire. Cet événement ne reçoit pas de confirmation et, en conséquence, aucune indication d'erreur ou de réussite n'est attendue en réponse.

routeTrace EVENT	*Temporisation* = 0	*Classe* = 4	*Code* = 00000010

2.1.2.1 Information d'événement

2.1.2.1.1 Réussite (*success*)

Contient la trace des codes de points (un ou plusieurs) des points sémaphores traversés en cas d'aboutissement correct.

Paramètre	*Code*
success	10100000

2.1.2.1.2 Boucle détectée (*detectedLoop*)

Contient les codes de points (trois ou plus) de la boucle en cas de détection de boucle.

Paramètre	*Code*
detectedLoop	10100001

2.1.2.1.3 Longueur d'itinéraire excessive (*excessiveLengthRoute*)

Contient la totalité de l'itinéraire lorsqu'un itinéraire de longueur excessive est trouvé (dépassement de seuil).

Paramètre	*Code*
excessiveLengthRoute	10100010

2.1.2.1.4 Destination inconnue (*unknownDestination*)

Aucune information supplémentaire n'est nécessaire si la destination n'est pas connue, puisque le paramètre "demande d'information" n'était pas présent dans la demande CNF-ACTION de test d'itinéraire.

2.1.2.1.5 Itinéraire inaccessible (*routeInaccessible*)

Contient le code de point du nœud au niveau duquel l'itinéraire était inaccessible.

Paramètre	*Code*
routeInaccessible	10000100

2.1.2.1.6 Erreur de traitement (*processingFailure*)

Aucune autre information n'est nécessaire en cas d'erreur de traitement.

Paramètre	*Code*
processingFailure	10000101

2.1.2.1.7 Point sémaphore initiateur inconnu (*unknownInitiatingSP*)

Contient le code de point du nœud qui détecte le point sémaphore initiateur inconnu.

Paramètre	*Code*
unknownInitiatingSP	10000110

2.1.2.1.8 Expiration de temporisation (*timerExpired*)

Contient le ou les codes de point des nœuds, desquels aucun résultat de l'action "test d'itinéraire" n'a été reçu.

Paramètre	*Code*
timerExpired	10100111

2.1.2.1.9 Le point sémaphore n'est pas un point de transfert sémaphore (*sPNotAnSTP*)

Contient la liste des points sémaphores traversés pour atteindre un point sémaphore qui reçoit un message MRVT, alors qu'il ne dispose pas de la fonction de transfert du sous-système MTP.

Une valeur de type de défaillance égale à "le point sémaphore n'est pas un point de transfert sémaphore" peut également signifier que le point sémaphore intermédiaire qui reçoit un message MRVT n'a pas l'autorisation de transférer des messages reçus de l'émetteur MRVT, comme le spécifient les étiquettes MTP de points sémaphores d'initiateur de test et de destinataire du test.

Paramètre	*Code*
sPNotAnSTP	10101000

2.1.2.1.10 Le nombre maximal de test MRVT est déjà atteint (*maxNrMRVTestsAlready*)

Ce compte rendu est utilisé par le point sémaphore recevant le message MRVT si le nombre maximal n_T de tests MRV est déjà actif au niveau du point sémaphore. Ceci donne lieu à un compte rendu d'erreur de traitement, voir 2.1.2.1.6, si le message MRVT de demande ("test d'itinéraire") ne contenait pas le paramètre "demande d'information".

2.1.3 Nouvelle trace d'itinéraire (*routeTraceNew*)

Ce compte rendu est utilisé si l'action "test d'itinéraire" incitatrice contient un paramètre "demande d'information".

routeTraceNew EVENT	*Temporisation* = 0	*Classe* = 4	*Code* = 00000100

2.1.3.1 Information d'événement

2.1.3.1.1 Réussite (*success*)

En cas de fin correcte, la trace des codes de point des points sémaphores traversés (un ou plusieurs) figure dans le paramètre "liste de codes de point" (copié à partir du paramètre "codes de points traversés" de l'action "test d'itinéraire").

Le paramètre "liste de priorités d'itinéraire" est copié à partir de l'action "test d'itinéraire", si cette dernière est présente et demandée dans le paramètre "demande d'information".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

2.1.3.1.2 Boucle détectée (*detectedLoop*)

Les codes de points des points sémaphores (trois ou plus) de la boucle figurent dans le paramètre "liste de codes de point" en cas de détection de boucle.

Le paramètre "liste de priorités d'itinéraire" est copié à partir de l'action "test d'itinéraire", si cette dernière est présente et demandée dans le paramètre "demande d'information".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

2.1.3.1.3 Longueur d'itinéraire excessive (*excessiveLengthRoute*)

Si cette erreur se manifeste, la totalité de l'itinéraire est copiée du paramètre "codes de points traversés" de l'action "test d'itinéraire" vers le paramètre "liste de codes de point".

Le paramètre "liste de priorités d'itinéraire" est copié à partir de l'action "test d'itinéraire", si cette dernière est présente et demandée dans le paramètre "demande d'information".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

2.1.3.1.4 Destination inconnue (*unknownDestination*)

Ce paramètre est équivalent au paramètre "destination inconnue" du 2.1.2.1.4. Si le paramètre "demande d'information" de l'action "test d'itinéraire" incitatrice le demande, le paramètre "codes de points traversés" de l'action "test d'itinéraire" est copiée dans le paramètre "liste de codes de point".

Le paramètre "liste de priorités d'itinéraire" est copié à partir de l'action "test d'itinéraire", si cette dernière est présente et demandée dans le paramètre "demande d'information".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

2.1.3.1.5 Itinéraire inaccessible (*routeInaccessible*)

Si cet événement ne rend compte que d'un seul point sémaphore inaccessible, le code de point correspondant est placé dans le paramètre "code de point".

Si cet événement rend compte de plus d'un point sémaphore inaccessible (et qu'en conséquence l'action "test d'itinéraire" incitatrice a indiqué que l'initiateur est susceptible de l'accepter), la liste de ces points sémaphores inaccessibles est placée dans le paramètre "liste de codes de point"

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" initiatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

2.1.3.1.6 Erreur de traitement (*processingFailure*)

L'événement renvoie un compte rendu d'erreur de traitement si le test ne peut être effectué en raison de conditions locales. Ceci inclut le rejet de l'action "test d'itinéraire" par le sous-système SCCP ou par le gestionnaire de transaction au niveau d'un point sémaphore distant.

Si le paramètre "demande d'information" de l'action "test d'itinéraire" était présent avec le bit 0 positionné en 1, le code de point du point sémaphore au niveau duquel le test a échoué est placé dans le paramètre "code de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

2.1.3.1.7 Point sémaphore initiateur inconnu (*unknownInitiatingSP*)

Le code de point du point sémaphore qui détecte l'initiateur inconnu est renvoyé dans le paramètre "code de point".

Si le résultat de l'action incitatrice "test d'itinéraire" contenait un paramètre "copie de données", celui-ci est copié dans le paramètre "copie de données" du paramètre "nouvelle trace d'itinéraire".

2.1.3.1.8 Expiration de temporisation (*timerExpired*)

Les codes de point des points sémaphores desquels aucun résultat n'a été obtenu pour les actions "test d'itinéraire" sont placés dans le paramètre "liste de codes de point".

2.1.3.1.9 Le point sémaphore n'est pas un point de transfert sémaphore (*sPNotAnSTP*)

Cette erreur se manifeste si le point sémaphore intermédiaire ne dispose pas de la fonction de point sémaphore de transfert ou s'il est connu qu'il n'est pas autorisé à transférer des messages de l'initiateur du test vers la destination de test.

Le paramètre "codes de points traversés" de l'action "test d'itinéraire" incitatrice est copié dans le paramètre "liste de codes de point".

S'il est présent dans l'action "test d'itinéraire", le paramètre "liste de priorités d'itinéraire" est copié dans le paramètre "demande d'information".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

2.1.3.1.10 Itinéraire indirect (*indirectRoute*)

Ce compte rendu est utilisé si le test d'itinéraire direct a été demandé et que le point sémaphore recevant le message MRVT ne disposait pas d'un itinéraire direct vers l'initiateur du test à partir de l'émetteur du message MRVT. L'identité du message MRVT instigateur est placée dans le paramètre "code de point", l'identité du point sémaphore détectant l'absence d'itinéraire direct est le code de point d'origine de l'étiquette MTP du message MRVR.

2.1.3.1.11 Le nombre maximal de tests MRVT est déjà atteint (*maxNrMRVTestsAlready*)

Ce compte rendu est utilisé par le point sémaphore recevant le message MRVT si le nombre maximal de tests MRV n_T est déjà actif sur le point sémaphore.

Si le paramètre "demande d'information" de l'action "test d'itinéraire" était présent avec le bit 0 positionné sur 1, le code de point du point sémaphore au niveau duquel le test a échoué est placé dans le paramètre "code de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3 Sous-système SCCP

3.1 Elément ASE de test de vérification d'acheminement du sous-système SCCP (SRVT)

Ces fonctions spécifiques du test SRVT sont définies dans la clause 3.2.2/Q.753. Le test de vérification d'acheminement lancé à l'origine du test génère une primitive OM-CONFIRMED-ACTION envoyée par l'utilisateur OMASE vers l'élément OMASE, contenant comme paramètre la commande "test d'itinéraire". Si la trace des itinéraires est demandée, ou si une faute se manifeste, la primitive OM-EVENT-REPORT est invoquée, avec le paramètre "trace d'itinéraire", au niveau de l'initiateur du test depuis l'élément OMASE.

Le paramètre "test d'itinéraire" est spécifié au moyen de l'objet CNF-ACTION définie dans la Figure 3, le paramètre "trace d'itinéraire" est spécifié au moyen de la classe EVENT définie dans la Figure 3.

La classe d'objets indique les tables de traduction de l'appellation globale du SCCP, et l'instance d'objet contient l'indicateur d'appellation globale et le titre global testé. L'indicateur d'appellation globale est codé comme défini dans l'indicateur d'adresse du sous-système SCCP. L'action "test d'itinéraire" (SRVT) utilise le message BEGIN avec un résultat (SRVA) renvoyé dans un message END. L'événement "trace d'itinéraire" (SRVR) utilise un message BEGIN avec une fin prédéterminée.

3.1.1 Action test d'itinéraire (*testRouteAction*)

L'action "test d'itinéraire" est invoquée pour lancer un test de vérification d'acheminement du sous-système SCCP. Au niveau du nœud d'origine, cette invocation est demandée par l'Administration au moyen de l'utilisateur MIS ou d'une interface locale à travers le processus de gestion du sous-système OMAP et l'utilisateur OMASE. Dans les nœuds suivants, l'action est demandée d'une manière implicite par la réception d'une invocation de l'action "test d'itinéraire". Une réponse positive indique la réussite du test au point où il a été invoqué, et d'une manière implicite à tous les points où le test à été invoqué. Une indication de faute est renvoyée pour indiquer que le test a échoué au niveau de ce nœud ou d'un nœud suivant.

TestRoute CNF-ACTION	*Temporisation* = T2	*Classe* = 1	*Code* = 00000001

3.1.1.1 Arguments de l'action d'itinéraire de test

3.1.1.1.1 Point sémaphore initiateur (*initiatingSP*)

Le paramètre "point sémaphore initiateur" identifie l'initiateur du test. Il est du type "code de point".

Paramètre	*Code*
initiatingSP	10000000
Contenu	
le bit 0 contient le premier bit du code de point le bit 1 contient le deuxième bit du code de point, etc.	

3.1.1.1.2 Trace demandée (*traceRequested*)

Le paramètre "trace demandée" indique qu'une trace de tous les itinéraires utilisés pour atteindre la destination doit faire l'objet d'un compte rendu à l'initiateur. Il est du type BOOLEAN.

Paramètre	*Code*
traceRequested	10000001
Contenu	*Signification*
"Vrai" (= 1)	une trace a été demandée, renvoi de l'information de trace en cas de succès et de faute.
"Faux" (= 0)	pas de trace demandée, renvoi de l'information de trace uniquement en cas de faute.

3.1.1.1.3 Seuil (*threshold*)

L'initiateur définit un seuil maximal pour le nombre de points sémaphores de traduction (TSP, *translation signalling points*) qui peuvent être traversés au cours du test (y compris l'initiateur s'il est un nœud de relais du SCCP). Ceci facilite la détection d'itinéraires de longueur anormale. Ce seuil indique un nombre entier de points sémaphores et son type est INTEGER.

Paramètre	*Code*
threshold	10000010

3.1.1.1.4 Codes de points traversés (*pointCodesTraversed*)

Chaque point sémaphore de traduction traversé ajoute son code de point à la liste des codes de points traversés. Ceci facilite la détection de boucles et constitue également une information utile en cas de défaillance ou lorsqu'une trace d'itinéraire est demandée. Il s'agit d'une liste de codes de point dont le type est "liste de codes de point". Cette liste de codes de point peut être vide.

Paramètre	*Code*
pointCodesTraversed	10100011
Contenu	
suite de codes de points, étiquetés "Code de point" dont le contenu indique le code de point exact	

3.1.1.1.5 Indicateur de forme (*formIndicator*)

L'indicateur de forme identifie la forme du message SRVT message, c'est-à-dire une demande, une vérification ou une comparaison. Son type est INTEGER, avec les valeurs définies ci-dessous.

Paramètre	*Code*
formIndicator	10000100
Contenu	
valeur 0 = comparaison valeur 1 = pas de comparaison	

3.1.1.1.6 Acheminement de retour du sous-système MTP demandé (*mtpBackwardRoutingRequested*)

Le paramètre "acheminement de retour du sous-système MTP demandé" indique si un acheminement de retour vers le code de point d'origine est exigé pour la réussite du test. Son type est BOOLEAN.

Paramètre	*Code*
mtpBackwardRoutingRequested	10000101
Contenu	
"Vrai" (= 1) acheminement demandé "Faux" (= 0) acheminement non demandé	

3.1.1.1.7 Appellation globale de l'initiateur du test (*testInitiatorGT*)

L'appellation globale de l'initiateur du test identifie l'indicateur d'appellation globale et l'appellation globale de l'initiateur. Son type est OCTET STRING.

Paramètre	*Code*
testInitiatorGT	10000110
Contenu	
octet 1 bits 3 à 6 = indicateur d'appellation globale octet 2, 3, ... = appellation globale de l'initiateur	

3.1.1.1.8 Code de point de destination (*destinationPC*)

Le paramètre "code de point de destination" identifie le point de code de destination (PPC ou TPC). Son type est "code de point".

Paramètre	*Code*
destinationPC	10000111
Contenu	
le bit 0 contient le premier bit du point de code le bit 1 contient le deuxième bit du point de code, etc.	

3.1.1.1.9 Numéro de sous-système de destination (*destinationSSN*)

Le paramètre "numéro de sous-système de destination" identifie le numéro de sous-système de destination. Son type est OCTET STRING.

Paramètre	*Code*
destinationSSN	10001000
Contenu	
le bit 0 contient le premier bit du sous-système de destination. le bit 1 contient le deuxième bit du sous-système de destination, etc.	

3.1.1.1.10 Le point de code de destination de secours (*backupDPC*)

Le paramètre "point de code de destination de secours" identifie le point de code de destination (SPC) de secours. Son type est "code de point".

Paramètre	*Code*
backupDPC	10001001
Contenu	
le bit 0 contient le premier bit du code point. le bit 1 contient le deuxième bit code point, etc.	

3.1.1.1.11 Numéro de sous-système de secours (*backupSSN*)

Le paramètre "numéro de sous-système de secours" identifie le numéro de sous-système de secours. Son type est OCTET STRING.

Paramètre	*Code*
backupSSN	10001010
Contenu	
le bit 0 contient le premier bit du numéro de sous-système. le bit 1 contient le deuxième bit du numéro de sous-système, etc.	

3.1.1.1.12 Appellation globale d'origine (*originalGT*)

Le champ "appellation globale d'origine" ne figure dans un message de test SRVT que si une traduction de l'appellation globale de l'adresse appelée produit une appellation globale de remplacement, ou a déjà produit une telle adresse.

Dans un tel cas, le champ à émettre dans un message SRVT est le suivant:

i) si le message SRVT à émettre n'a pas la forme "compare", et si le test est lancé par la réception d'un message SRVT contient un paramètre "appellation globale d'origine", auquel cas ce champ est copié; ou

ii) dans tous les autres cas, le paramètre "appellation globale d'origine" émis est l'appellation globale de l'adresse d'appelé du message SRVT avant traduction.

Le champ est utilisé comme appellation globale de l'adresse d'appelé dans tout message SRVR et, pour la forme "compare" du message SRVT, par le point sémaphore de traduction homologue qui le reçoit, afin de vérifier si sa traction fournit l'appellation globale reçue dans le champ d'adresse d'appelé du message SRVT de type compare.

Le type du paramètre "appellation globale d'origine" est "appellation globale".

Paramètre	*Code*
originalGT	10001011
Contenu	
octet 1 bits 3 à 6 = indicateur de traduction globale octet 2, 3,... = traduction globale d'origine	

3.1.1.1.13 Traduction globale entrante (*inputGT*)

Le paramètre traduction globale entrante", utilisé uniquement dans la forme "compare" du message SRVT, identifie le test fait sur l'indicateur de traduction globale et la traduction globale avant leur traduction au niveau d'un point sémaphore de traduction. Son type est OCTET STRING.

Paramètre	*Code*
inputGT	10010000
Contenu	
octet 1 bits 3 a 6 = indicateur de traduction globale octet 2, 3,... = traduction globale entrée, à destination du point sémaphore de traduction	

3.1.1.1.14 Demande d'information (*infoRequest*)

Ce paramètre optionnel, pouvant uniquement être inséré par le point sémaphore initiateur du test, indique que l'initiateur peut reconnaître des messages SRVR résultant d'un événement de type "nouvelle trace d'itinéraire". Le paramètre "demande d'information" indique quelles sont les informations exigées si un message SRVR doit être émis vers l'initiateur. Il peut également indiquer quels sont les paramètres devant être mis à jour lorsque les messages SRVT traversent le réseau. Les valeurs effectives peuvent être "code de point" (bit 0 = 1) et "liste de codes de point" (bit 1).

Paramètre	*Code*
infoRequest	10001101
Contenu	
chaîne de bits contenant une ou plusieurs des valeurs indiquées	

3.1.1.1.15 Retour de paramètres inconnus (*returnUnknownParams*)

Ce paramètre facultatif peut uniquement être inséré par le point sémaphore initiateur du test (et seulement si le paramètre "demande d'information" est également présent). Il indique quels sont les paramètres SRVT qu'un nœud suivant devra renvoyer, si ce successeur ne reconnaît pas les paramètres en question. Les paramètres SRVT non reconnus doivent être recopiés dans le (nouveau) message SRVR ("nouvelle trace d'itinéraire") si le nœud suivant a l'occasion de renvoyer un message SRVR, ou dans le paramètre "copie de données" d'un message SRVA s'il ne connaît pas l'initiateur. Le bit 0 dans le paramètre "retour de paramètres inconnus" indique un paramètre SRVT avec une valeur d'étiquette égale à 15, le bit 1 un paramètre SRVT avec une valeur d'étiquette égale à 16, et ainsi de suite.

Paramètre	*Code*
returnUnknownParams	10001110
Contenu	
chaîne de bits contenant une ou plusieurs des valeurs indiquées	

3.1.1.2 Résultats de l'action

Une indication de retour positive ne contient pas de paramètre.

3.1.1.3 Erreurs de l'action

Les erreurs spécifiques sont des erreurs possibles pouvant survenir pendant un test et qui sont propres à ce test. Ces erreurs spécifiques existent en plus des erreurs déjà identifiées dans le service OM-CONFIRMED-ACTION et apparaissent comme paramètres de l'erreur de défaillance de traitement.

3.1.1.3.1 Défaillance (*failure*)

L'erreur spécifique "défaillance" indique une condition de défaillance totale lorsqu'une traduction n'a pas pu être faite ou s'est faite d'une manière incorrecte. Elle sera utilisée le plus souvent comme indication de défaillance à partir du point qui détecte l'erreur et n'invoque pas d'action "test d'itinéraire" ultérieure. L'erreur spécifique "défaillance" véhicule un paramètre permettant d'indiquer la condition d'erreur qui cause la défaillance. Ce paramètre "type de défaillance" est représenté par une chaîne binaire. Le deuxième paramètre à utiliser conjointement indique l'erreur "point sémaphore initiateur non connu". Le paramètre "trace émise" indique si un événement "trace d'itinéraire" a été invoqué pour rendre compte d'une information d'itinéraire. Il est nécessaire d'indiquer cette information pour cette erreur, parce que le nœud qui détecte l'erreur ne peut émettre d'événement "trace d'itinéraire" et c'est, en conséquence, le nœud précédent qui doit le faire. Le paramètre "trace émise" est facultatif et du type BOOLEAN.

Erreur spécifique	*Code*
failure	00000001

Paramètre	*Code*
failureType	10000000

Paramètre	*Code*
traceSent	10000001
Contenu	*Signification*
"Vrai"	l'information de trace a été émise
"Faux"	l'information de trace n'a pas été émise

3.1.1.3.2 Réussite partielle (*partialSuccess*)

Cette indication est fournie lorsqu'au moins une invocation de l'action "test d'itinéraire" a échoué et qu'au moins une autre a réussi, au moins en partie. Dans ce cas, tout type d'erreur qui s'est manifestée

sera noté et émis dans la réponse finale. Le format et le contenu du paramètre "réussite partielle" sont les mêmes que ceux du paramètre "défaillance".

Erreur spécifique	*Code*
partialSuccess	00000010

3.1.2 Evénement de trace d'itinéraire (*routeTrace*)

L'événement "trace d'itinéraire" rend compte d'une information de trace. L'information de trace se constitue de un ou plusieurs codes de point, donnant par exemple la liste totale des codes de point traversés par un itinéraire. Cet événement est invoqué soit sur demande explicite du nœud initiateur (indiqué par "trace demandée", voir 3.1.1.1.2), soit par l'événement "défaillance" pour tout point situé sur l'itinéraire. Cet événement ne reçoit pas de confirmation et, en conséquence, aucune indication d'erreur ou de réussite n'est attendue en réponse.

routeTrace EVENT	*Temporisation* = 0	*Classe* = 4	*Code* = 00000010

3.1.2.1 Information d'événement

3.1.2.1.1 Réussite (*success*)

Contient la trace des codes de points (un ou plusieurs) des nœuds de relais du sous-système SCCP traversés en cas d'aboutissement correct.

Paramètre	*Code*
success	10100000

3.1.2.1.2 Boucle détectée (*detectedLoop*)

Contient les codes de points (trois ou plus) de la boucle en cas de détection de boucle.

Paramètre	*Code*
detectedLoop	10100001

3.1.2.1.3 Longueur d'itinéraire excessive (*excessiveLengthRoute*)

Contient la totalité de l'itinéraire lorsqu'un itinéraire de longueur excessive est trouvé (dépassement de seuil).

Paramètre	*Code*
excessiveLengthRoute	10100010

3.1.2.1.4 Destination inconnue (*unknownDestination*)

Aucune information supplémentaire n'est nécessaire si la destination n'est pas connue. Pour le message SRVT, ceci se réfère au cas dans lequel il n'existe pas de données de traduction pour l'indicateur de traduction globale et la traduction globale.

Paramètre	*Code*
unknownDestination	10000011

3.1.2.1.5 Itinéraire inaccessible (*routeInaccessible*)

Contient le code de point du nœud au niveau duquel l'itinéraire était inaccessible.

Paramètre	*Code*
routeInaccessible	10000100

3.1.2.1.6 Erreur de traitement (*processingFailure*)

Aucune autre information n'est nécessaire en cas d'erreur de traitement.

Paramètre	*Code*
processingFailure	10000101

3.1.2.1.7 Point sémaphore initiateur inconnu (*unknownInitiatingSP*)

Contient le code de point du nœud qui détecte le point sémaphore initiateur inconnu.

Paramètre	*Code*
unknownInitiatingSP	10000110

3.1.2.1.8 Expiration de temporisation (*timerExpired*)

Contient le ou les codes de point des codes de points, desquels aucun résultat de l'action "test d'itinéraire" n'a été reçu.

Paramètre	*Code*
timerExpired	10100111

3.1.2.1.9 Point sémaphore non valide (*wrongSP*)

Contient la liste complète des points sémaphores de traduction vers le point sémaphore non valide.

Paramètre	*Code*
wrongSP	10101000

3.1.2.1.10 Traduction primaire incorrecte (*incorrectTranslation-Primary*)

Contient la liste complète des points sémaphores de traduction vers la destination primaire incorrecte.

Paramètre	*Code*
incorrectTranslation-Primary	10101001

3.1.2.1.11 Traduction secondaire incorrecte (*incorrectTranslation-Secondary*)

Contient la liste complète des points sémaphores de traduction vers la destination secondaire incorrecte.

Paramètre	*Code*
incorrectTranslation-Secondary	10101010

3.1.2.1.12 Traduction intermédiaire incorrecte (*incorrectTranslation-Intermediate*)

Contient la liste complète des points sémaphores de traduction vers le point intermédiaire incorrect.

Paramètre	*Code*
incorrectTranslation-Intermediate	10101011

3.1.2.1.13 Destination primaire non valide (*notPrimaryDestination*)

Contient la liste complète des points sémaphores de traduction vers la destination primaire non valide.

Paramètre	*Code*
notPrimaryDestination	10101100

3.1.2.1.14 Destination secondaire non valide (*notSecondaryDestination*)

Contient la liste complète des points sémaphores de traduction vers la destination secondaire non valide.

Paramètre	*Code*
notSecondaryDestination	10101101

3.1.2.1.15 Destination primaire non reconnue (*notRecognizedPrimary*)

Contient la liste complète des points sémaphores de traduction vers la destination secondaire.

Paramètre	*Code*
notSecondaryDestination	10101110

3.1.2.1.16 Destination secondaire non reconnue (*notRecognizedSecondary*)

Contient la liste complète des points sémaphores de traduction traversés par l'itinéraire vers la destination primaire.

Paramètre	*Code*
notRecognizedSecondary	10101111

3.1.2.1.17 Problème d'acheminement (*routingProblem*)

Contient la liste complète des points sémaphores de traduction traversés par l'itinéraire vers le problème d'acheminement éventuel. Ce cas se présente lorsque le code de point fourni par la traduction n'est pas reconnu.

Paramètre	*Code*
routingProblem	10110000

3.1.3 Evénement de nouvelle trace d'itinéraire (*routeTraceNew Event*)

Ce compte rendu est utilisé si l'action "test d'itinéraire" incitatrice contenait un paramètre "demande d'information".

routeTraceNew EVENT	*Temporisation* = 0	*Classe* = 4	*Code* = 00000100

3.1.3.1 Information d'événement (*Event information*)

L'information contenue est la même que pour l'action "trace d'itinéraire", mais avec le paramètre "résultat" indiquant le résultat et avec les données associées au résultat présentes dans les paramètres optionnels. Le paramètre "copie de données" éventuellement présent contient les paramètres de l'action "test d'itinéraire" incitatrice qui n'ont pas été compris et dont le renvoi a été demandé par le paramètre "retour de paramètres inconnus" dans l'action "test d'itinéraire". Le résultat supplémentaire "nombre maximal de tests SRVR déjà atteint" (*maxNrSRVRTestsAlready*) est défini à l'heure actuelle pour l'action "nouvelle trace d'itinéraire".

3.1.3.1.1 Réussite (*success*)

En cas de fin correcte, la trace des codes de point des points sémaphores traversés (un ou plusieurs) figure dans le paramètre "liste de codes de point" (copié à partir du paramètre "codes de points traversés" de l'action "test d'itinéraire").

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.2 Boucle détectée (*detectedLoop*)

Les codes de points des points sémaphores (trois ou plus) de la boucle figurent dans le paramètre "liste de codes de point" en cas de détection de boucle.

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.3 Longueur d'itinéraire excessive (*excessiveLengthRoute*)

Si cette erreur se manifeste, la totalité de l'itinéraire est copiée du paramètre "codes de points traversés" de l'action "test d'itinéraire" vers le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.4 Destination inconnue (*unknownDestination*)

Si le paramètre "demande d'information" de l'action "test d'itinéraire" incitatrice en fait la demande, le paramètre "codes de points traversés" de l'action "test d'itinéraire" est copié dans liste de codes de point.

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.5 Itinéraire inaccessible (*routeInaccessible*)

Si cet événement ne rend compte que d'un seul point sémaphore inaccessible, le code de point correspondant est placé dans le paramètre "code de point".

Si cet événement rend compte de plus d'un point sémaphore inaccessible (et qu'en conséquence l'action "test d'itinéraire" incitatrice a indiqué que l'initiateur est susceptible de l'accepter), la liste de ces points sémaphores inaccessibles est placée dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.6 Erreur de traitement (*processingFailure*)

L'événement renvoie un compte rendu d'erreur de traitement si le test ne peut être effectué en raison de condition locales. Ceci inclut le rejet de l'action "test d'itinéraire" par le sous-système SCCP ou par le gestionnaire de transaction au niveau d'un point sémaphore distant.

Si le paramètre "demande d'information" de l'action "test d'itinéraire" était présent avec le bit 0 positionné en 1, le code de point du point sémaphore au niveau duquel le test a échoué est placé dans le paramètre "code de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.7 Point sémaphore initiateur inconnu (*unknownInitiatingSP*)

Le code de point du point sémaphore qui détecte l'initiateur inconnu est renvoyé dans le paramètre "code de point".

Si le résultat de l'action incitatrice "test d'itinéraire" contenait un paramètre "copie de données", celui-ci est copié dans le paramètre "copie de données" du paramètre "nouvelle trace d'itinéraire".

3.1.3.1.8 Expiration de temporisation (*timerExpired*)

Les codes de point des points sémaphores desquels aucun résultat n'a été obtenu pour les actions "test d'itinéraire" sont placés dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.9 Point sémaphore faux (*wrongSP*)

Le paramètre "codes de points traversés" de l'action "test d'itinéraire" incitatrice est copié dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.10 Traduction incorrecte (*incorrectTranslation-Primary*)

La liste complète des points sémaphores de traduction vers la destination primaire incorrecte est copiée dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.11 Traduction incorrecte (*incorrectTranslation-Secondary*)

La liste complète des points sémaphores de traduction vers la destination secondaire incorrecte est copiée dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.12 Traduction incorrecte (*incorrectTranslation-Intermediate*)

La liste complète des points sémaphores de traduction vers le point intermédiaire incorrect est copiée dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.13 Destination primaire non valide (*notPrimaryDestination*)

La liste complète des points sémaphores de traduction vers la destination primaire non valide est copiée dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.14 Destination secondaire non valide (*notSecondaryDestination*)

La liste complète des points sémaphores de traduction vers la destination secondaire non valide est copiée dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.15 Destination primaire non reconnue (*notRecognizedPrimary*)

La liste complète des points sémaphores de traduction vers la destination secondaire est copiée dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.16 Destination secondaire non reconnue (*notRecognizedSecondary*)

La liste complète des points sémaphores de traduction traversés par l'itinéraire vers la destination primaire est copiée dans le paramètre "liste de codes de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.17 Problème d'acheminement (*routingProblem*)

La liste complète des points sémaphores de traduction traversés par l'itinéraire contenant le problème d'acheminement éventuels est copiée dans le paramètre "liste de codes de point". Ceci survient lorsque le code de point fourni par la traduction n'est pas reconnu.

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

3.1.3.1.18 Le nombre maximal de tests SRV est déjà atteint (*maxNrSRVTestsAlready*)

Ce compte rendu est utilisé par le point sémaphore recevant le message SRVT si le nombre maximal de tests n_{SRVT} est déjà actif au niveau du point sémaphore.

Si le paramètre "demande d'information" de l'action "test d'itinéraire" était présent avec le bit 0 positionné sur 1, le code de point du point sémaphore au niveau duquel le test a échoué est placé dans le paramètre "code de point".

Les paramètres éventuellement non reconnus dans l'action "test d'itinéraire" incitatrice sont copiés dans le paramètre "copie de données" si l'action "test d'itinéraire" contient un paramètre "retour de paramètres inconnus" qui en fait la demande.

4 Gestion de circuit

4.1 Elément ASE de test de validation de circuit (CVT, *circuit validation test*)

L'élément ASE de test de validation de circuit fournit les services accédés au moyen de l'action OM-CONFIRMED-ACTION, comme décrit dans la Figure 3. Il utilise une instance de la classe d'objets de gestion de circuit décrite dans la Recommandation Q.751. La classe d'objets gérés de base (*BaseManagedObjectClass*) indique la valeur "cvt-Cic-Tables-1992", et l'instance d'objet géré de base (*BaseManagedObjectInstance*) identifie l'information *CktGrpInfo* (information de groupe de circuit identifiant l'identificateur prédéfini pour le circuit et son groupe, ayant fait l'objet d'un accord entre les commutateurs situés aux extrémités du groupe de circuits), le code CIC étant connu du point sémaphore émetteur.

4.1.1 Action de confirmation de validation de demande de test (*cktValidTest CnfAction*)

La demande de test de validation de circuit et le renvoi de la réponse de validation de circuit qui en résulte est mappée dans une action de confirmation. Cette action est la demande de test à l'extrémité distante.

cktValidTest CNF-ACTION	*Temporisation* = T_c	*Classe* = 1	*Code* = 00000001

Se référer au 4.2/Q.753 pour ce qui est des valeurs possibles de T_c et de la temporisation.

```
cktValidTest CNF-ACTION ::=
        {
        ACTIONARG SEQUENCE {
                requestingSP        RequestingSP,
                timer               Timer OPTIONAL,
                …}

        ACTIONRESULT                Success
        SPECIFICERRORS              { failure }
        CODE                3
        }
    -- Temporisateur = Tc , class = 1
```

4.1.2 Arguments de l'action

Le paramètre "point sémaphore demandeur" (*requestingSP*) est le code de point du point sémaphore qui lance la procédure de test, il est du type chaîne d'octet comme indiqué ci-dessous.

```
RequestingSP ::= OCTET STRING
```

4.1.3 Résultats de l'action

Les résultats de l'action sont renvoyés en cas de réussite dans un composant de retour de résultat. Le contenu des deux paramètres doit être défini dans les procédures CVT.

```
Success ::= SEQUENCE
                {
                cktGrpInfo   [0] IMPLICIT CktGrpInfo,
                cICName      [1] IMPLICIT OCTET STRING OPTIONAL,
                …}
```

Il convient de noter que le paramètre CktGrpInfo est défini comme étant du type OCTET STRING.

4.1.4 Erreur spécifique

Le paramètre erreur spécifique indique un échec et le motif de l'échec. Le contenu des deux paramètres doit être défini dans les procédures CVT.

```
Failure ::= SPECIFIC-ERROR
        {
        PARAMETER SEQUENCE      {cktGrpInfo  [0] IMPLICIT CktGrpInfo,
                cICName [1] IMPLICIT OCTET STRING OPTIONAL,
                …}
        CODE        3
        }
```

Il convient de noter que le paramètre CktGrpInfo est défini comme étant du type OCTET STRING.

Les motifs d'échec de test CVT sont les suivants:

a) code CIC non attribué à l'extrémité locale;

b) données d'extrémité locale du circuit non valides;

c) pas de tonalité valide reçue à l'extrémité locale;

d) expiration de la temporisation générale de test T_c avant la réception du message CVR.

e) message CVR reçu avant que la synchronisation ait été faite dans la configuration binaire de test;

f) la temporisation T_c expire avant que la synchronisation ait été faite dans la configuration binaire de test;

g) la configuration binaire de test est encore en cours de réception au moment de l'expiration de la temporisation T_c;

h) la configuration binaire de test est encore en cours de réception au moment de la réception du message CVR;

i) une tonalité est en cours de réception au moment de l'expiration de la temporisation T_c;

j) une tonalité est en cours de réception lorsque le message est reçu;

k) les codes CIC des extrémités locale et distante ne correspondent pas (vérification faite par l'extrémité locale au moment de la réception du message CVR);

l) le message CVR reçu indique une erreur:

- pas de code CIC attribué au niveau de l'extrémité distante;
- données d'extrémité distante du circuit non valides;
- caractéristiques de groupe non disponibles à l'extrémité distante;

m) erreur non spécifiée.

5 Capacités de gestionnaire de transaction

Appellent une étude ultérieure.

La réponse au sein du sous-système OMAP à une diffusion locale des messages N-STATE et N-PCSTATE appelle également une étude ultérieure.

6 Généralités

6.1 Objets et définitions

Le sous-système OMAP effectue des tests sur des objets tels que les tables d'acheminement du sous-système MTP et du sous-système SCCP. Ces objets sont décrits dans ce document comme "classe d'objets" et sont identifiés par un identificateur d'objet spécifiant la présente Recommandation et le type d'objet. Cette structure est indiquée ci-dessous pour les identificateurs d'objets mtp-Routing-Tables, sccp-Routing-Tables, et cvt-Cic-Tables.

oMAP	OBJECT IDENTIFIER :: = { itu-t recommendation q 754 }
mtp-Routing-Tables-1992	OBJECT IDENTIFIER :: = { oMAP 0 }
sccp-Routing-Tables-1992	OBJECT IDENTIFIER :: = { oMAP 1 }
cvt-Cic-Tables-1992	OBJECT IDENTIFIER :: = { oMAP 5 }

La valeur de la classe d'objets est égale à 0011857200 (hexadécimal) pour les tables d'acheminement du sous-système MTP, 0011857201 (hexadécimal) pour les tables d'acheminement du sous-système SCCP et 0011857205 (hexadécimal) pour les tables de codes de circuits du test CVT. Prière de se référer aux Recommandations X.680 et X.690.

Les Tableaux 1 et 2 donnent les primitives d'exploitation et de maintenance, la Figure 1 indique les opérations du sous-système OMAP découlant du protocole CMIP (ISO/CEI 9596) et la Figure 3 donne une syntaxe abstraite de l'élément OMASE.

Opérations définies à l'heure actuelle	
0	eventReport
7	confirmedAction

Figure 1/Q.754 – Opérations OMAP découlant du protocole CMIP

6.2 Primitives et procédures du protocole OMASE

6.2.1 Généralités

Le protocole OMASE utilise le service de gestion de transaction tel qu'il est défini dans la Recommandation Q.771. L'identité d'invocation et l'identité de dialogue correspondent à celles définies pour le service de gestion de transaction.

L'élément OMASE est modélisé par une machine de protocole (appelée OMPM par la suite). L'abréviation APDU désigne dans le texte qui suit une unité de données de protocole d'application et fait référence au contenu des primitives transmises entre l'élément OMASE et le gestionnaire de transaction.

La Figure A.1 présente le modèle incluant le gestionnaire de transaction et le sous-système SCCP, la machine OMPM réside dans l'élément OMASE. La Figure A.2 donne un exemple d'instances particulières des primitives dans un test MRV (mais sans primitive OM-EVENT-REPORT).

6.2.2 Service OM-EVENT-REPORT

6.2.2.1 Primitive du service

La primitive OM-EVENT-REPORT utilisée entre l'utilisateur OMASE et l'élément OMASE est présentée dans le Tableau 1.

L'événement spécifique qui s'est manifesté est interprété dans le contexte de la classe d'objets spécifiée.

Tableau 1/Q.754 – Paramètres de la primitive OM-EVENT-REPORT

Nom du paramètre	dem/ind
CallingPartyAddress	M
CalledPartyAddress	M
DialogueID	M
InvokelD	M
ManagedObjectClass	M
ManagedObjectInstance	M
EventType	M
EventTime	O
EventInfo	O

Définitions de paramètres

Adresse de l'appelant (*CallingPartyAddress*): telle qu'elle est définie dans l'adresse appelante au 2.2/Q.711.

Adresse de l'appelé (*CalledPartyAddress*): telle qu'elle est définie dans l'adresse appelée, 2.2/Q.711. Les adresses ci-dessus servent à identifier le sous-système OMAP aux niveaux des points sémaphores appelants et appelés. Elles peuvent être toutes deux, pour le message MRVT, sous la forme d'un code de point avec en plus un numéro de sous-système (OMAP). Pour un test SRVT, elles se trouvent dans une forme convenant au type d'acheminement du sous-système SCCP mis en œuvre dans le test.

Identificateur de dialogue (*DialogueID*): tel qu'il est défini dans les Recommandations Q.771 à Q.775. Il correspond à l'identificateur de transaction défini dans la Recommandation Q.772.

Identificateur d'invocation (*InvokeID*): tel qu'il est défini dans la Recommandation Q.772.

Classe d'objets gérés (*ManagedObjectClass*): identifie la classe d'objets pour laquelle est défini cet événement.

Instance d'objet géré (*ManagedObjectInstance*): identifie l'instance d'objet auquel s'applique le compte rendu d'événement.

Type d'événement (*EventType*): spécifie l'événement donné dont rend compte l'instance d'objet.

Date de l'événement (*EventTime*): spécifie la date de création de l'événement.

Information d'événement (*EventInfo*): fournit une information supplémentaire propre à l'événement.

6.2.2.2 Procédure de compte rendu d'événement

6.2.2.2.1 Réception d'une demande OM-EVENT-REPORT

Les procédures de compte rendu d'événement sont démarrées par la primitive de demande OM-EVENT-REPORT. Lorsque ceci se produit, la machine OMPM construit une unité APDU demandant l'opération "compte rendu d'événement" (*eventReport*) et la transmet au moyen des services fournis par les primitives TC-INVOKE et TC-BEGIN

La primitive de demande TC-INVOKE contient les valeurs et paramètres suivants:

- identificateur de dialogue – défini par l'utilisateur OMASE;
- identificateur d'invocation – défini par l'utilisateur OMASE;
- opération – positionnée sur "compte rendu d'événement";
- classe – positionnée sur 4;
- paramètres – correspondent aux paramètres qui suivent le mot clé PARAMETER dans la définition de l'opération "compte rendu d'événement". La valeur du paramètre "type d'événement" spécifie quelle est l'action à exécuter – elle doit indiquer "trace d'itinéraire" pour les procédures qui sont définies à l'heure actuelle;
- temporisation – Positionné sur 0 pour le message MRVT et le message SRVT.

La primitive de demande TC-BEGIN contient les valeurs et paramètres suivants:

- adresse de destination – Telle qu'elle est reçue dans le paramètre "adresse de l'appelé" de la primitive de demande OM-EVENT-REPORT;
- adresse d'origine – Telle qu'elle est reçue dans le paramètre "adresse de l'appelant" de la primitive de demande OM-EVENT-REPORT;
- identificateur de dialogue – Tel que présent dans la primitive TC-INVOKE.

La primitive de demande N-UNITDATA émise en dernier lieu vers le sous-système SCCP à la suite de la réception de ces primitives de gestion de transaction contient l'ensemble de paramètres de commande de séquence indiquant "séquence garantie" et le paramètre option de retour doit être positionné pour indiquer "rejet du message en cas d'erreur". Voir 2.2.2/Q.711.

La machine OMPM termine le dialogue après la transmission de l'unité APDU, au moyen d'une primitive de demande TC-END contenant les paramètres "identificateur de dialogue" et "terminaison", ce dernier indiquant "fin prédéfinie".

6.2.2.2.2 Réception de la primitive TC-BEGIN avec une indication TC-INVOKE

La machine OMPM émettra une primitive d'indication OM-EVENT-REPORT lorsqu'elle reçoit une unité APDU de forme correcte, faisant une demande d'opération "compte rendu d'événement" pour les primitives d'indication TC-BEGIN et TC-INVOKE. La machine OMPM rejette l'unité APDU si celle-ci n'a pas une forme correcte.

La machine OMPM termine le dialogue avec une primitive de demande TC-END contenant les paramètres "identificateur de dialogue" et "terminaison", ce dernier indiquant "fin prédéfinie".

6.2.2.2.3 Réception d'une primitive TC-BEGIN avec une indication TC-L-REJECT

La machine OMPM émet, dans ce cas, une primitive de demande TC-END contenant les paramètres "identificateur de dialogue" et "terminaison", ce dernier indiquant "fin prédéfinie".

6.2.2.2.4 Réception d'une indication TC-P-ABORT

La machine OMPM ignore, dans ce cas, la primitive TC-P-ABORT.

6.2.3 Service OM-CONFIRMED-ACTION

6.2.3.1 Primitive du service

Le service OM-CONFIRMED-ACTION est présenté dans le Tableau 2. L'action spécifique à exécuter est interprétée dans le contexte de la classe d'objets spécifiée. Le service se fait avec confirmation (un compte rendu de réussite ou d'échec est émis dans tous les cas).

Tableau 2/Q.754 – Service OM-CONFIRMED-ACTION

Nom du paramètre	Dem/Ind	Rep/Conf
CallingPartyAddress	M	M
CalledPartyAddress	M	M
DialogueID	M	M
InvokeID	M	M
AccessControl	O	–
BaseManagedObjectClass	M	–
BaseManagedObjectInstance	M	–
ActionInfo	M	–
ActionResult	–	M[a)]
ActionError	–	M[b)]
Timer	M[c)]	–

[a)] Obligatoire dans le composant de retour de l'action (éventuellement vide).

[b)] Obligatoire dans le composant de retour d'erreur.

[c)] Ce paramètre ne figure que dans la primitive de demande.

Définitions de paramètres

Adresse de l'appelant (*CallingPartyAddress*): voir le Tableau 1.

Adresse de l'appelé (*CalledPartyAddress*): voir le Tableau 1.

Identificateur de dialogue (*DialogueID*): mis en correspondance par le sous-système TCAP avec un identificateur de transaction défini dans la Recommandation Q.772.

Identificateur d'invocation (*InvokeID*): tel qu'il est défini dans la Recommandation Q.772.

Contrôle d'accès (*AccessControl*): information à utiliser comme entrée pour les fonctions de contrôle d'accès.

Classe d'objets gérés de base (*BaseManagedObjectClass*): identifie la classe d'objets pour laquelle est définie cette action.

Instance d'objet géré de base (*BaseManagedObjectInstance*): identifie l'instance d'objet pour laquelle doit être effectuée l'action.

Information de l'action (*ActionInfo*): séquence d'un type ActionType et d'un argument optionnel ActionInfoArg. Le type ActionType, défini par la macro CNF-ACTION, spécifie une action particulière devant être effectuée sur l'instance de l'objet.

Résultat de l'action (*ActionResult*): ce champ contient le résultat en cas de réussite de l'action.

Erreur de l'action (*ActionError*): ce champ indique une information d'erreur ou un statut de problème lorsque l'action n'a pas totalement réussi.

Temporisation (*Timer*): ce paramètre contient la valeur particulière pour la durée de la temporisation d'attente d'une réponse. Il est positionné sur T_1 pour le test MRVT, T_2 pour le test SRVT, ou T_c pour le test CVT.

Les valeurs sont indiquées dans la Recommandation Q.753.

6.2.3.2 Procédures pour l'action confirmée (*confirmedAction*)

6.2.3.2.1 Réception de la primitive de demande OM-CONFIRMED-ACTION

La procédure "action confirmée" (*confirmedAction*) est lancée par la réception de la primitive de demande OM-CONFIRMED-ACTION. La machine OMPM construit dans ce cas une unité APDU demandant l'opération "action confirmée" et transmet cette unité APDU au moyen des services fournis par les primitives TC-INVOKE et TC-BEGIN.

La primitive de demande TC-INVOKE contient les valeurs et paramètres suivants:

- opération – prend la valeur "action confirmée";
- classe – positionné sur 1;
- paramètres – correspondent aux paramètres qui suivent le mot clé "PARAMETER" dans la définition de l'opération "action confirmée". La valeur du paramètre "test d'itinéraire" est obtenue par dérivation à partir de l'objet CNF-ACTION de la forme locale de l'identificateur de type d'action du type d'action contenu dans l'information d'action;
- temporisation – copiée à partir du paramètre correspondant de la primitive de demande OM-CONFIRMED-ACTION.
- les identificateurs de dialogue et d'invocation sont copiés à partir de la primitive de demande OM-CONFIRMED-ACTION.

La primitive de demande TC-BEGIN utilise les paramètres suivants:

- identificateur de dialogue – Tel qu'il figure dans la primitive TC-INVOKE.
- adresse de destination – Telle qu'elle est reçue dans le paramètre "adresse de l'appelé" de la primitive de demande OM-CONFIRMED-ACTION.
- adresse d'origine – Telle qu'elle est reçue dans le paramètre "adresse de l'appelant" de la primitive de demande OM-CONFIRMED-ACTION.

La primitive de demande N-UNITDATA, émise en dernier lieu vers le sous-système SCCP à la suite de la réception de ces primitives de gestion de transaction, doit contenir le paramètre de contrôle de séquence positionné sur "séquence non garantie" et le paramètre option de retour doit être positionné sur "rejet du message en cas d'erreur". Voir 2.2.2/Q.711.

6.2.3.2.2 Réception d'une primitive TC-BEGIN avec une indication TC-INVOKE

La machine OMPM émet dans ce cas une primitive d'indication OM-CONFIRMED-ACTION lorsqu'elle reçoit une unité APDU de forme correcte, faisant une demande d'opération "action confirmée" à destination de l'utilisateur OMASE.

La machine OMPM ignore les indications du gestionnaire de transaction si l'unité APDU n'a pas une forme correcte.

Un mécanisme propre à la mise en œuvre comparable à celui défini au 3.3.4/Q.774 doit traiter tout problème local.

Lorsque l'unité APDU contient d'autres paramètres, la machine OMPM transmet ces paramètres de façon transparente à l'utilisateur OMASE.

6.2.3.2.3 Réception d'une primitive de réponse OM-CONFIRMED-ACTION

La primitive de réponse OM-CONFIRMED-ACTION peut contenir soit le paramètre "résultat de l'action", soit le paramètre "erreur de l'action".

Le paramètre "résultat de l'action" indique la réussite de l'exécution de l'opération et la machine OMPM émet une primitive de demande TC-RESULT-L. Dans le cas d'un test CVT, les paramètres suivants figurent dans la primitive TC-RESULT-L:

- opération – positionné sur la valeur "action confirmée".
- paramètres – correspondent au paramètre "réussite" du résultat de l'action pour le test CVT.

La présence du paramètre "résultat de l'action" indique l'échec de l'opération et la machine OMPM émet une primitive de demande TC-U-ERROR contenant les paramètres suivants:

- "erreur" – contient la valeur d'erreur adéquate prise dans l'ensemble défini après le mot clé "ERRORS" dans la définition de l'opération;
- "paramètres" – correspondent aux paramètres définis après le mot clé "PARAMETER" dans la définition de l'erreur.

Le résultat de l'opération est transmis par la machine OMPM au moyen de l'émission d'une primitive de demande TC-END contenant les paramètres "identificateur de dialogue" et "terminaison", ce dernier indiquant "fin de base".

La primitive de demande N-UNITDATA émise en dernier lieu vers le sous-système SCCP à la suite de la réception de ces primitives de demande de gestion de transaction doit contenir le paramètre de contrôle de séquence positionné pour indiquer "séquence non garantie" et le paramètre option de retour doit être positionné pour indiquer "rejet du message en cas d'erreur". Voir 2.2.2/Q.711.

6.2.3.2.4 Réception d'une primitive TC-END avec une indication TC-RESULT-L

Lorsqu'elle reçoit une unité APDU de forme correcte, la machine OMPM émet vers l'utilisateur OMASE une primitive de confirmation OM-CONFIRMED-ACTION avec le paramètre "résultat de l'action" (contenant l'indicateur de dialogue).

La machine OMPM ignore les primitives TC[3] si l'unité APDU n'a pas une forme correcte.

6.2.3.2.5 Réception d'une primitive TC-END avec une indication TC-U-ERROR

Lorsqu'elle reçoit une unité APDU de forme correcte, la machine OMPM émet vers l'utilisateur OMASE une primitive de confirmation OM-CONFIRMED-ACTION avec le paramètre "erreur de l'action" (contenant l'indicateur de dialogue).

La machine OMPM ignore les primitives TC[3] si l'unité APDU n'a pas une forme correcte.

6.2.3.2.6 Réception d'une primitive d'indication TC-L-CANCEL

Ce cas se présente lorsque la temporisation d'invocation expire.

La machine OMPM émet une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique "défaillance" pour l'action CNF-ACTION, et le paramètre "type de défaillance" indiquant "expiration de temporisation".

---

[3] L'utilisation de la temporisation générale de garde dans l'utilisateur de l'élément OMASE au niveau du noeud initiateur du test permet au test d'échouer d'une manière ordonnée dans ces circonstances.

La machine OMPM termine le dialogue avec une primitive de demande TC-END avec le paramètre "terminaison" indiquant "fin prédéfinie".

6.2.3.2.7 Réception d'une primitive TC-BEGIN ou TC-END avec une indication TC-L-REJECT

La Figure 2a illustre le cas de réception d'une primitive TC-BEGIN avec une indication TC-L-REJECT.

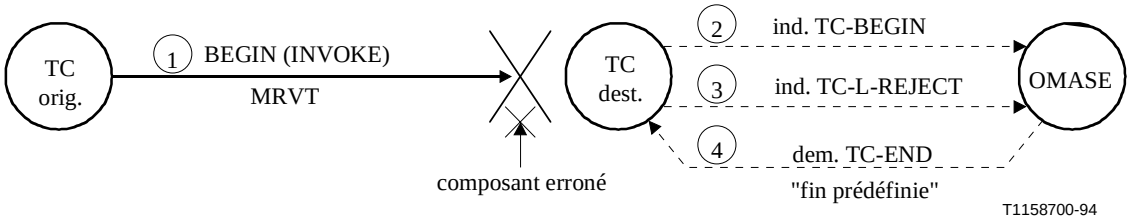

Figure 2a/Q.754

Si elle reçoit une primitive d'indication TC-L-REJECT avec une indication TC-BEGIN, la machine OMPM termine le dialogue en émettant une primitive de demande TC-END avec le paramètre "terminaison" indiquant "fin prédéfinie".

Si la machine OMPM reçoit une primitive d'indication TC-L-REJECT avec une indication TC-END, elle émet une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique "défaillance" de l'action CNF-ACTION et, si "test d'itinéraire" a été invoqué, le paramètre "type de défaillance" de la primitive de confirmation indique "erreur de traitement". Ceci est illustré par la Figure 2b.

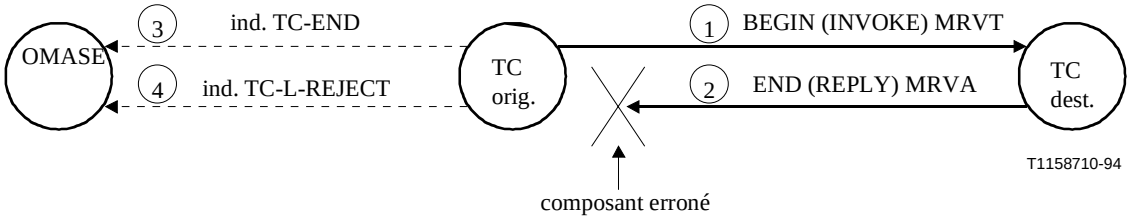

Figure 2b/Q.754

6.2.3.2.8 Réception d'une primitive TC-END avec une indication TC-R-REJECT

La machine OMPM émet dans ce cas une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique "défaillance" de l'action CNF-ACTION et, si "test d'itinéraire" a été invoqué, le paramètre "type de défaillance" de la primitive de confirmation indique "erreur de traitement".

6.2.3.2.9 Réception d'une primitive d'indication TC-P-ABORT

Ce cas est illustré par les deux diagrammes de la Figure 2c.

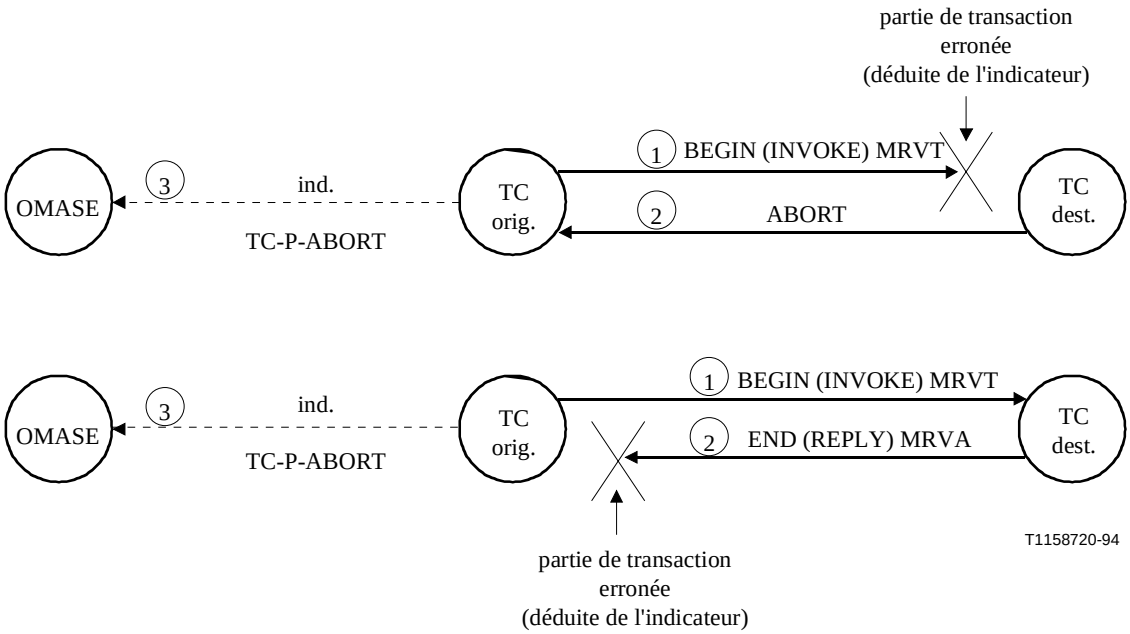

Figure 2c/Q.754

La machine OMPM émet une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique "failure" de l'action CNF-ACTION et, si "test d'itinéraire" a été invoqué, avec le paramètre "type de défaillance" de la primitive de confirmation indiquant "erreur de traitement".

6.2.3.2.10 Réception d'une primitive TC-NOTICE

La machine OMPM émet dans ce cas une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique "failure" de l'action CNF-ACTION et, si "test d'itinéraire" a été invoqué, avec le paramètre "type de défaillance" de la primitive de confirmation indiquant "erreur de traitement".

Définitions d'erreurs

Les définitions des deux services OM font référence à un certain nombre d'erreurs qui sont définies dans le présent sous-paragraphe.

Définitions

Classe d'objets non valide (*noSuchObjectClass*): la classe d'objets contenue dans l'unité APDU d'invocation n'est pas reconnue par l'extrémité réceptrice.

Instance d'objet non valide (*noSuchObjectInstance*): la classe d'objets contenue dans l'unité APDU d'invocation est reconnue, mais il n'existe pas, au niveau de l'extrémité réceptrice, d'objet correspondant appartenant à cette classe.

Accès refusé (*accessDenied*): l'accès à la ressource n'est pas autorisé.

Erreur de traitement (*processingFailure*): une erreur s'est manifestée en cours de traitement d'une action ou d'un événement spécifique. Les indicateurs de faute sont spécifiques de l'action ou de l'événement.

Action invalide (*noSuchAction*): ce type d'action n'est pas reconnu ou n'est pas pris en charge par l'extrémité réceptrice.

Argument invalide (*noSuchArgument*): cet argument n'est pas reconnu ou n'est pas pris en charge par l'extrémité réceptrice.

Valeur d'argument invalide (*invalidArgumentValue*): la valeur de l'argument ne convient pas à l'extrémité réceptrice.

6.3 Syntaxe abstraite du protocole d'élément OMASE

Voir la Figure 3.

```
-- Protocole d'élément OMASE --
OMASE { itu-t(0) recommendation q 754 omase(0) version2(2) }

DEFINITIONS EXPLICIT TAGS::=
BEGIN
-- Définitions du sous-système TCAP --
EXPORTS EVERYTHING;

-- les objets d'information OPERATION et ERROR définis ici sont respectivement équivalents aux MACRO
-- figurant dans les messages TCAPMessages
-- {ccitt recommendation q 773 modules(2) messages(1) version2(2) } de la Rec. Q.773 (1993) --

OPERATION ::= CLASS
{
        &ArgumentType       OPTIONAL,
        &ResultType         OPTIONAL,
        &Errors             ERROR OPTIONAL,
        &Linked             OPERATION OPTIONAL,
        &operationCode      Code UNIQUE OPTIONAL
}
WITH SYNTAX
{
        [PARAMETER    &ArgumentType]
        [RESULT       &ResultType]
        [ERRORS       &Errors]
        [LINKED       &Linked]
        [CODE         &operationCode]
}

ERROR ::= CLASS
{
        &ParameterType      OPTIONAL,
        &errorCode          Code UNIQUE OPTIONAL
}
WITH SYNTAX
{
        [PARAMETER    &ParameterType]
        [CODE         &errorCode]
}

Code ::= CHOICE
{
        localValue      INTEGER.
        GlobalValue     OBJECT IDENTIFIER
}
```

Figure 3/Q.754 (feuillet 1 de 11) – Syntaxe formelle des services OMASE

```
-- Opérations OMASE --

eventReport OPERATION ::=
{
PARAMETER eventReportArgument          EventReportArgument

CODE       localValue:0
}

confirmedAction OPERATION ::=
{
PARAMETER actionArgument          ActionArgument

RESULT     actionResult          ActionResult
ERRORS     { accessDenied | invalidArgumentValue |
           noSuchAction | noSuchArgument |
           noSuchObjectClass | noSuchObjectInstance |
           processingFailure }
CODE       localValue:7
}
```

Figure 3/Q.754 (feuillet 2 de 11) – Syntaxe formelle des services OMASE

-- Les définitions d'erreurs du service OM sont les suivantes: --

```
noSuchObjectClass       ERROR ::=
        {
        PARAMETER       ObjectClass
        CODE            localValue:0
        }

noSuchObjectInstance    ERROR ::=
        {
        PARAMETER       ObjectInstance
        CODE            localValue:1
        }

accessDenied            ERROR ::=
        {
        CODE            localValue:2
        }

noSuchAction            ERROR ::=
        {
        PARAMETER       NoSuchAction
        CODE            localValue:9
        }

processingFailure       ERROR ::=
        {
        PARAMETER       ProcessingFailure -- optionnel --
        CODE            localValue:10
        }

noSuchArgument          ERROR ::=
        {
        PARAMETER       NoSuchArgument
        CODE            localValue:14
        }

invalidArgumentValue    ERROR ::=
        {
        PARAMETER       InvalidArgumentValue
        CODE            localValue:15
        }
```

Figure 3/Q.754 (feuillet 3 de 11) – Syntaxe formelle des services OMASE

-- Les définitions de types auxiliaires sont données ci-après: --

```
ActionArgument      ::= SEQUENCE    {
                        COMPONENTS OF       BaseManagedObjectId,
                        accessControl       [5] AccessControl OPTIONAL,
                        actionInfo          [12] IMPLICIT ActionInfo }

ActionInfo          ::= SEQUENCE    {
                        actionType          [3] IMPLICIT CNF-ACTION.&operationCode,
                        actionInfoArg       [4] CNF-ACTION. &ActionArgType(@actionType)
                                            OPTIONAL}

ActionResult        ::= SEQUENCE    {
                        managedObjectClass
                        ObjectClass OPTIONAL,
                        managedObjectInstance   ObjectInstance OPTIONAL,
                        currentTime             [5] IMPLICIT GeneralizedTime OPTIONAL,
                        actionReply             [6] IMPLICIT ActionReply OPTIONAL }

ActionTypeId        ::= CHOICE      {
                        -- forme globale... --
                        localForm               [3] IMPLICIT CNF-ACTION }

BaseManagedObjectId ::= SEQUENCE    {
                        baseManagedObjectClass      ObjectClass,
                        baseManagedObjectInstance   ObjectInstance }

EventReportArgument ::= SEQUENCE    {
                        managedObjectClass      ObjectClass,
                        managedObjectInstance   ObjectInstance,
                        eventTime               [5] IMPLICIT GeneralizedTime OPTIONAL,
                        eventType               [7] IMPLICIT EVENT. &operationCode
                        eventInfo               [8] EVENT. &EventInfo Type (@eventType)
                                                OPTIONAL}

EventTypeId         ::= CHOICE      {
                        -- forme globale... --
                        localForm               [7] IMPLICIT EVENT }

ActionReply         ::= SEQUENCE    {
                        actionType              [3] IMPLICIT CNF-ACTION.
                                                &operationCode,
                        actionReplyInfo         [4]
                                        CNF-ACTION.&ActionResultType(@actionType)
                        }

AccessControl       ::= EXTERNAL
```

-- La syntaxe du type AccessControl doit être compatible avec celle définie dans le protocole CMIP codée comme X.209.

Figure 3/Q.754 (feuillet 4 de 11) – Syntaxe formelle des services OMASE

```
InvalidArgumentValue    ::= CHOICE {
                                actionValue         [0] IMPLICIT ActionInfo,
                                eventValue          [1] IMPLICIT SEQUENCE {
                                eventType           [7] IMPLICIT EVENT.&operationCode,
                                eventInfo           [8] EVENT.&EventInfoType(@eventType)
                                                        OPTIONAL }}

NoSuchAction            ::= SEQUENCE    { managedObjectClass    ObjectClass,
                                          actionType            ActionTypeId }

NoSuchArgument          ::= CHOICE      {
                                actionId                        [0] IMPLICIT SEQUENCE {
                                        managedObjectClass      ObjectClass OPTIONAL,
                                        actionType              ActionTypeId },
                                eventId                         [1] IMPLICIT SEQUENCE {
                                        managedObjectClass      ObjectClass OPTIONAL,
                                        eventType               EventTypeID }}

ObjectClass             ::= CHOICE      {
                                globalForm                      [0] IMPLICIT OBJECT
                                                                IDENTIFIER,

                                -- ... --
                                }

ObjectInstance          ::= CHOICE      {
                                -- ... --
                                nonSpecificForm                 [3] IMPLICIT OCTET STRING,
                                -- ... --
                                }

ProcessingFailure       ::= SEQUENCE    {
                                managedObjectClass              ObjectClass OPTIONAL,
                                managedObjectInstance           ObjectInstance OPTIONAL,
                                specificErrorInfo               [5] IMPLICIT SpecificErrorInfo }

SpecificError           ::= INTEGER     -- définie par classe d'objets --

SpecificErrorInfo       ::= SEQUENCE    {
                                errorType           [0] IMPLICIT SpecificError.&errorCode,
                                errorParm           [1] IMPLICIT SPECIFIC-ERROR. &ProcessingError
                                                    ParmType(@errorType) OPTIONAL }

Timer                   ::= INTEGER     -- secondes --
```

Figure 3/Q.754 (feuillet 5 de 11) – Syntaxe formelle des services OMASE

```
-- Des catégories sont définies par classe d'objets pour les comptes rendus spécifiques d'événements.
-- Les protocoles utilisés peuvent être décrits au moyen de l'objet
-- EVENT MACRO ci-dessous

    EVENT ::= CLASS
    {
        &EventInfoType          OPTIONAL,
        &operationCode          INTEGER UNIQUE OPTIONAL
    }
    WITH SYNTAX
    {
        [EVENTINFO              &EventInfoType]
        [CODE                   &operationCode]
    }

-- Des catégories sont définies par classe d'objets pour les comptes rendus spécifiques d'événements.
-- Les protocoles  utilisés peuvent être décrits au moyen de l'objet
-- CNF-ACTION INFORMATION OBJECT ci-dessous --

    CNF-ACTION ::= CLASS
    {
        &ActionArgType          OPTIONAL,
        &ActionResultType       OPTIONAL,
        &SpecificErrors         SPECIFIC-ERROR OPTIONAL,
        &operationCode          INTEGER UNIQUE OPTIONAL
    }
    WITH SYNTAX
    {
        [ACTIONARG              &ActionArgType]
        [ACTIONRESULT           &ActionResultType]
        [SPECIFICERRORS         &SpecificErrors]
        [CODE                   &operationCode]
    }

-- Les erreurs spécifiques d'une action ou d'un événement sont définies au moyen de la macro
-- SPECIFIC-ERROR ci-dessous --

        SPECIFIC-ERROR ::= CLASS
        {
            &ProcessingErrorParmType        OPTIONAL,
            &errorCode                      INTEGER UNIQUE OPTIONAL
        }
        WITH SYNTAX
        {
            [PARAMETER          &ProcessingErrorParmType]
            [CODE               &errorCode]
        }
```

Figure 3/Q.754 (feuillet 6 de 11) – Syntaxe formelle des services OMASE

```
-- les expressions propres à l'élément OMASE sont données ci-dessous --

testRoute CNF-ACTION ::=
        {
                ACTIONARG SEQUENCE{
                                        initiating SP           [0] IMPLICIT PointCode,
                                        traceRequested          [1] IMPLICIT BOOLEAN,
                                        threshold               [2] IMPLICIT INTEGER,
                                        pointCodesTraversed     [3] IMPLICIT PointCodeList,
                                        formindicator           [4] IMPLICIT Formindicator OPTIONAL,
-- formIndicator est exigé dans SRVT, mais n'est pas utilisé dans MRVT --

                                        mtpBackwardRoutingRequested
                                                                [5] IMPLICIT BOOLEAN OPTIONAL,
-- mtpBackwardRoutingRequested est exigé dans SRVT, mais pas dans MRVT --

                                        testInitiatorGT         [6] IMPLICIT GlobalTitle OPTIONAL,
                                        destinationPC           [7] IMPLICIT PointCode OPTIONAL,
                                        destinationSSN          [8] IMPLICIT SubsystemNumber
                                                                OPTIONAL,
                                        backupDPC               [9] IMPLICIT PointCode OPTIONAL,
                                        backupSSN               [10] IMPLICIT SubsystemNumber
                                                                OPTIONAL,
                                        originalGT              [11] IMPLICIT GlobalTitle OPTIONAL,
                                        inputGT                 [16] IMPLICIT GlobalTitle OPTIONAL,
-- les paramètres avec les étiquettes 4 à 12 ne peuvent être utilisés que dans SRVT, pas dans MRVT --

                                        routePriorityList       [12] IMPLICIT RoutePriorityList
                                                                        OPTIONAL,
-- routePriorityList  ne peut être utilisé que MRVT, et uniquement si infoRequest est présent --

                                        infoRequest             [13] IMPLICIT BIT STRING {
                                                                pointCode(0),
                                                                pointCodeList(1),
                                                                routePriorityList(2),
                                                                …} OPTIONAL,
-- infoRequest est utilisé pour indiquer que le nœud initiateur du test peut prendre en charge un message
-- RVR routeTraceNew, et demande également le renvoi de paramètres particuliers, s'il est émis. Ce paramètre
-- ne peut être inséré qu'au nœud initiateur, mais peut être copié dans des messages MRVT régénérés --

                                        infoRequest             [13] IMPLICIT BIT STRING {
                                                                pointCode(0),
                                                                pointCodeList(1),
                                                                routePriorityList(2),
                                                                …} OPTIONAL,
-- returnUnknownParams est utilisé pour indiquer quels sont les paramètres qu'un nœud ne peut
-- comprendre. Il doit être renvoyé dans un message RVR si un tel message est émis (ou dans le
-- champ copyData d'un message RVA si l'initiateur du test n'est pas connu). Le bit 0 représente un
-- paramètre RVT avec une valeur de marque de 15, le bit 1 d'un paramètre RVT avec une valeur de
-- marque de 16, etc. Pour éviter une confusion dans le champ copyData lors de la définiton
-- d'un nouveau paramètre dans le message RVR, la marque doit avoir la même valeur que celle
-- qu'elle avait dans le message RVT. Ce paramètre ne peut être présent que si infoRequest est présent --

                                                directRouteCheck [15] IMPLICIT BOOLEAN OPTIONAL,
-- directRouteCheck ne peut être utilisé que dans MRVT --

                … }

SPECIFICERRORS          { failure | partialSuccess }

CODE                    1

}

-- TC temporisation = T1 pour MRVT, = T2 pour SRVT, classe = 1
```

Figure 3/Q.754 (feuillet 7 de 11) – Syntaxe formelle des services OMASE

```
PointCode ::= OCTET STRING
PointCodeList ::= SEQUENCE OF PointCode
RoutePriorityList ::= SEQUENCE OF Priority

Priority ::= INTEGER{
                        unknown(0),
                        firstChoice(1),
                        secondChoice(2),
                        thirdChoice(3),
                        …} (0..255)

FormIndicator ::= INTEGER
        { compare (0),
        noCompare (1) } (0..1)

GlobalTitle ::= OCTET STRING
        -- le titre global est ici constitué de l'indicateur de titre global et du titre global
        -- SCCP; l'indicateur de titre global doit être codé exactement comme dans 3.4.1/Q.713, et
        -- le titre global, selon le cas, comme aux 3.4.2.1 à 3.4.2.4/Q.713. --

SubsystemNumber ::= OCTET STRING

failure     SPECIFIC-ERROR ::=
        {
        PARAMETER SEQUENCE          {failureType    [0] IMPLICIT FailureString,
                                    traceSent       [1] IMPLICIT BOOLEAN,
                                    copyData        [2] IMPLICIT CopyData OPTIONAL,
        -- copyData ne peut être présent que si failureType est égal à unknownInitiatingSP, si traceSent est ,
        -- "Faux" et si le message RVT incitateur contenait un paramètre requestInfo ou si
        -- returnUnknownParams se trouvait dans le message RVT. --
                                    … }
        CODE        1
        }

FailureString ::= BIT STRING
                { detectedLoop(0),
                excessiveLengthRoute(1),
                unknownDestination(2),
                routeInaccessible(3),
                processingFailure(4),
                unknownInitiatingSP(5),
                timerExpired(6),
                sPNotAnSTP(7),
        -- wrongSp est un synonyme de sPNotAnSTP utilisé dans SRVT. --
                incorrectTranslation-Primary (8),
                incorrectTranslation-Secondary (9),
                incorrectTranslation-Intermediate (10),
                notPrimaryDestination (11),
                notSecondaryDestination (12),
                notRecognizedPrimary (13),
                notRecognizedSecondary (14),
                routingProblem (15),
    -- les bits 8 à 15 ne peuvent être positionnés que dans SRVT, mais pas dans MRVT --
                maxNrMRVTestsAlready(16),
    -- maxNrSRVTestsAlready est un synonyme de maxNrMRVTestsAlready utilisé dans SRVT --
                indirectRoute(17),
    -- indirectRoute ne peut être positionné que dans MRVT, mais pas dans SRVT --
                ... }
    CopyData::=OCTET STRING
```

Figure 3/Q.754 (feuillet 8 de 11) – Syntaxe formelle des services OMASE

```
partialSuccess      SPECIFIC-ERROR ::=
        {
                PARAMETER SEQUENCE      {failureType        [0] IMPLICIT FailureString,
                                         traceSent          [1] IMPLICIT BOOLEAN,
                                         copyData           [4] IMPLICIT CopyData OPTIONAL,

        -- copyData ne peut être présent que si failureType est égal à unknownInitiatingSP, si traceSent est
        -- "Faux" et si le message RVT incitateur contenait un paramètre requestInfo ou si
        -- returnUnknownParams se trouvait dans le message RVT --

                                                 … }
                CODE        2
        }

routeTrace          EVENT ::=
                {
                EVENTINFO CHOICE {
                        success                 [0] IMPLICIT PointCodeList,
                        detectedLoop            [1] IMPLICIT PointCodeList,
                        excessiveLengthRoute    [2] IMPLICIT PointCodeList,
                        unknownDestination      [3] IMPLICIT NULL,
                        routeInaccessible       [4] IMPLICIT PointCode,
                        processingFailure       [5] IMPLICIT NULL,
                        unknownInitiatingSP     [6] IMPLICIT PointCode,
                        timerExpired            [7] IMPLICIT PointCodeList,
                        sPNotAnSTP          [8] IMPLICIT PointCodeList,
                -- wrongSP est un synonyme de sPNotAnSTP utilisé dans SRVT --

                incorrectTranslation-Primary        [9] IMPLICIT PointCodeList,
                incorrectTranslation-Secondary      [10] IMPLICIT PointCodeList,
                incorrectTranslation-Intermediate   [11] IMPLICIT PointCodeList,
                notPrimaryDestination               [12] IMPLICIT PointCodeList,
                notSecondaryDestination             [13] IMPLICIT PointCodeList,
                notRecognizedPrimary                [14] IMPLICIT PointCodeList,
                notRecognizedSecondary              [15] IMPLICIT PointCodeList,
                        routingProblem              [16] IMPLICIT PointCodeList
                -- les choix avec les étiquettes 9 à 16 ne peuvent être utilisés que dans SRVT. --

                        }
                CODE        2
                }

        -- TC Temporisateur = 0, classe = 4
```

Figure 3/Q.754 (feuillet 9 de 11) – Syntaxe formelle des services OMASE

```
routeTraceNew EVENT ::=
        {
                EVENTINFO SEQUENCE {
                        result                          [0] IMPLICIT ErrorTag,
                        pointCode                       [1] IMPLICIT PointCode OPTIONAL,
                        pointCodeList                   [2] IMPLICIT PointCodeList OPTIONAL,
                        routePriorityList               [3] IMPLICIT RoutePriorityList OPTIONAL,
                        copyData                        [4] IMPLICIT CopyData OPTIONAL,
        -- copyData permet, lorsque l'initiateur du test n'est pas connu, de copier tout paramètre conte nu
        -- dans un message RVA dans le message RVR sans modifier la structure de ce dernier. Il permet
        -- également, si l'initiateur de test en fait la demande, de renvoyer les nouveaux paramètres RVT
        -- facultatifs qui ne sont pas compris par le nœud qui produit le message RVR à partir du message
        -- RVA. Il convient de noter qu'un nouveau paramètre défini dans routeTraceNew doit avoir la même
        -- valeur que dans l'action testRoute s'il est défini dans cette dernière.
        -- Un message RVR distinct doit être émis pour chaque erreur détectée (les bits diagnostic d'erreur
        -- ne doivent pas être assemblés par un opérateur "OU" binaire). --
                                                        … }
                CODE        4

        }
        -- TC Temporisateur = 0, classe = 4
```

Figure 3/Q.754 (feuillet 10 de 11) – Syntaxe formelle des services OMASE

```
ErrorTag ::=INTEGER {
                success(0),
                detectedLoop(1),
                excessiveLengthRoute(2),
                unknownDestination(3),
                routeInaccessible(4)
                processingFailure(5)
                unknownInitiatingS(6),
                timerExpired(7),

        -- wrongSP est un synonyme de sPNotAnSTP utilisé dans SRVT --

        incorrectTranslation-Primary(9),
        incorrectTranslation-Secondary(10),
        incorrectTranslation-Intermediate(11),
        notPrimaryDestination(12),
        notSecondaryDestination(13),
        notRecognizedPrimary(14),
        notRecognizedSecondary(15),
                routingProblem(16)
        -- les valeurs de 9 à 16 ne s'appliquent que dans SRVT, mais pas dans MRVT --

                        maxNrMRVTestsAlready(17),
        -- maxNrSRVTestsAlready est un synonyme de maxNrMRVTestsAlready utilisé dans in SRVT --

                        indirectRoute(18),
        -- la valeur 18 ne s'applique que dans MRVT, mais pas dans SRVT --

                                … } (0..255)

END – Protocole OMASE --
```

Figure 3/Q.754 (feuillet 11 de 11) – Syntaxe formelle des services OMASE

ANNEXE A

Utilisation des interfaces de primitives

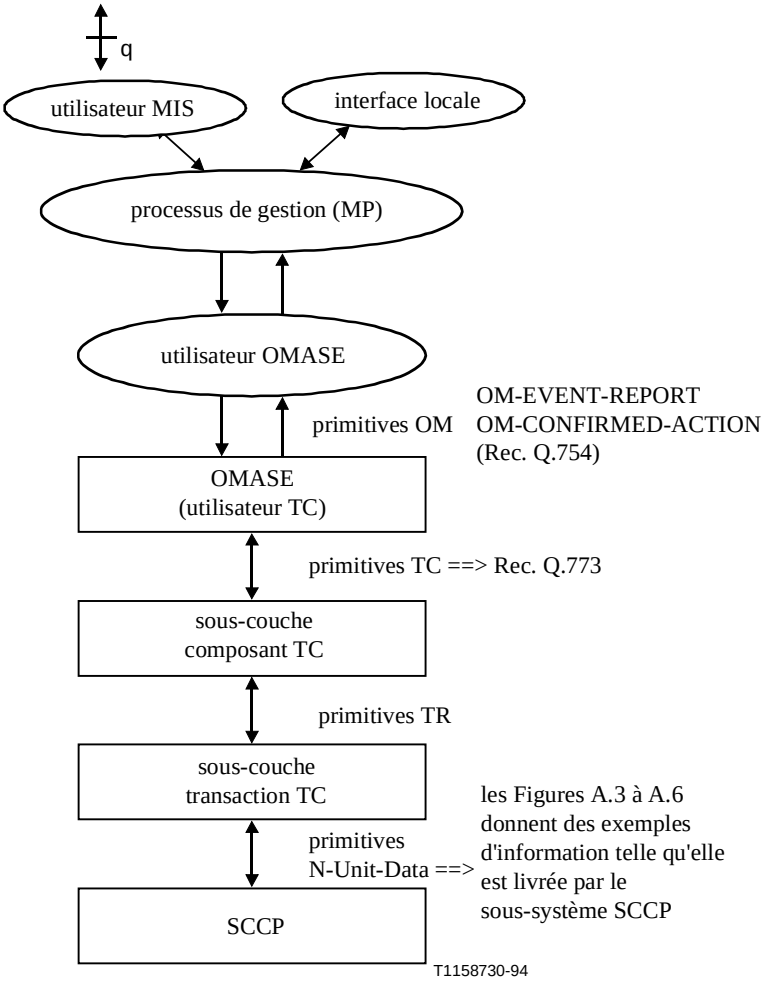

Figure A.1/Q.754 – Interfaces de primitives

La Figure A.2 illustre l'utilisation des primitives dans un test MRV. Lorsqu'il reçoit une demande d'émission de test MRVT issue du processus de gestion (MP – voir la Recommandation Q.753), l'utilisateur OMASE d'origine construit une demande OM-CONFIRMED-ACTION. La procédure se déroule ensuite jusqu'à l'étape numéro 5 comme indiqué par la succession de primitives et de messages. A cette étape, si le nœud n'est pas la destination testée, l'utilisateur OMASE qui reçoit la primitive d'indication OM-CONFIRMED-ACTION fait une demande OM-CONFIRMED-ACTION d'élément OMASE afin d'émettre des messages MRVT sur tous les itinéraires vers la destination testée figurant dans la table d'acheminement. Lorsque tous les messages MRVA sont reçus (vues par l'utilisateur OMASE comme des primitives de confirmation OM-CONFIRMED-ACTION), l'utilisateur OMASE émet les primitives de réponse OM-CONFIRMED-ACTION comme indiqué à l'étape numéro 6.

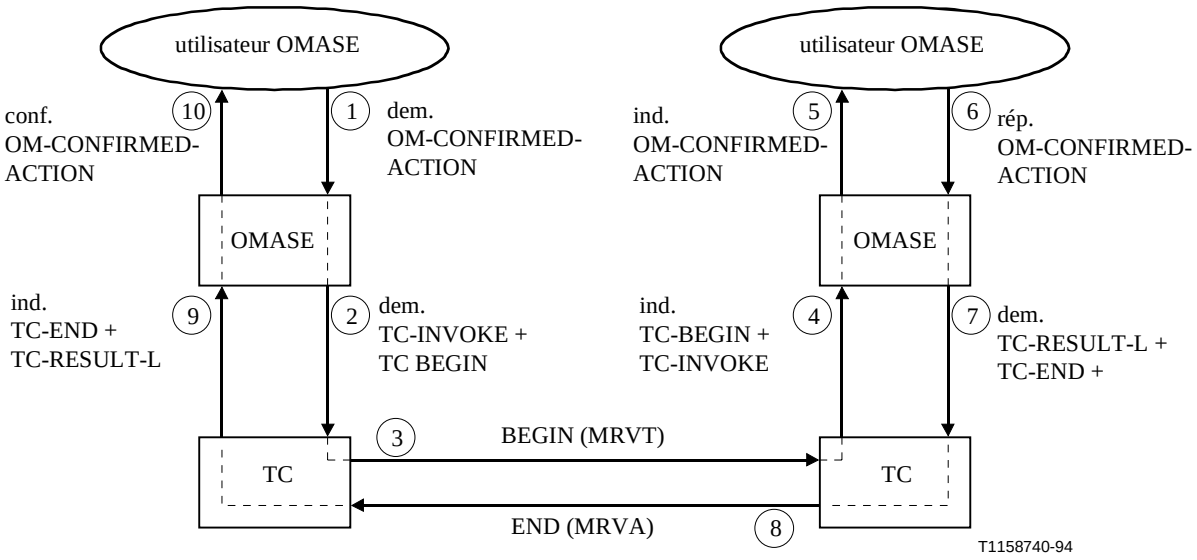

Figure A.2/Q.754 – Exemple d'utilisation des interfaces de primitives

Nom du champ	Codage binaire	Référence/Commentaire
marque de type de message	01100010	= début (Tableau 8/Q.773)
longueur du message	00110010	50 octets suivants: sous-système TC
marque d'identification de transaction	01001000	= Origine (Tableau 10/Q.773)
longueur	00000100	4 octets
valeur de l'identificateur de transaction	xxxxxxxx	sous-système TCAP
	xxxxxxxx	basé sur un dialogue
	xxxxxxxx	
	xxxxxxxx	
marque de partie de composant	01101100	(Tableau 14/Q.773)
longueur	00101010	tous les 42 octets suivants
marque de type de composant	10100001	= invocation (Tableau 19/Q.773)
longueur	00101000	tous les 40 octets suivants
marque d'identificateur de composant	00000010	= id. d'invocation (Tableau 20/Q.773)
longueur	00000001	1 octet
valeur d'identificateur d'invocation	xxxxxxxx	fournie par OMAP
marque de code opération	00000010	= local (Tableau 22/Q.773)
longueur	00000001	1 octet
code opération	00000111	= action confirmée (Figure 3/Q.754)
marque de séquence de paramètres	00110000	= marque de séq. (Tableau 23/Q.773)
longueur	00100000	32 octets suivants
marque de classe d'objets	10000000	Forme globale, Recs. X.711 et X.690
longueur	00000101	5 octets
valeur tables d'achemin. MTP	00000000	Rec. UIT-T
	00010001	q
	10000101	85 => 754
	01110010	72 =>
	00000000	tables d'acheminement MTP 1992
marque d'instance d'objet	10000011	forme non spécifique, Recs. X.711 et X.690
Longueur	00000010	2 octets
valeur d'instance d'objet	xxxxxxxx	destination testée (OMAP)
	xxxxxxxx	
marque d'information de l'action	10101100	Recs. X.711 et X.690
longueur	00010011	19 octets suivants
marque de type de l'action	10000011	forme locale X.711 et X.690
longueur	00000001	1 octet
CNF-ACTION	00000001	= test d'itinéraire (Figure 3/Q.754)
marque d'info. de l'action	10100100	Recs. X.711 et X.690
longueur	00001110	14 octets suivants
marque séq de paramètres	00110000	= marque de séq. (Tableau 23/Q.773)
longueur	00001100	12 octets suivants
marque de PS initiateur	10000000	Figure 3/Q.754, Rec. X.690
longueur	00000010	2 octets
valeur de PS initiateur	xxxxxxxx	initiateur du test (OMAP)
	xxxxxxxx	
marque de dem. de trace	10000001	Figure 3/Q.754, Rec. X.690
longueur	00000001	1 octet
valeur	00000001	= "Vrai"
marque de seuil	10000010	= seuil (Figure 3/Q.754)
longueur	00000001	1 octet
valeur de seuil	xxxxxxxx	fourni par OMAP
marque de code point Trav. traversé	10100011	Figure 3/Q.754
longueur	00000000	liste vide de codes de points

Figure A.3/Q.754 – Exemple de message MRVT fourni au sous-système SCCP

Nom du champ	Codage binaire	Référence/Commentaire
marque de type de message	01100100	= END (Tableau 8/Q.773)
longueur du message	00001101	13 octets suivants dans le sous-système gestionnaire de transaction
marque d'identification de transaction	01001001	= destination (Tableau 10/Q.773)
longueur	00000100	4 octets
valeur de l'identificateur de transaction	xxxxxxxx	comme dans BEGIN (message MRVT)
	xxxxxxxx	
	xxxxxxxx	
	xxxxxxxx	
marque de partie de composant	01101100	(Tableau 14/Q.773)
longueur	00000101	5 octets suivants
marque de type de composant	10100010	= retour de résultat (L) (Tableau 19/Q.773)
longueur	00000011	3 octets suivants
marque d'identificateur de composant	00000010	= id. d'invocation (Tableau 20/Q.773)
longueur	00000001	1 octet
valeur d'identificateur d'invocation	xxxxxxxx	comme dans le message MRVT (Corrélation)

Figure A.4/Q.754 – Exemple de message MRVA livré au sous-système SCCP (en cas de réussite)

Nom du champ	Codage binaire	Référence/Commentaire
marque de type de message	01100100	= END (Tableau 8/Q.773)
longueur du message	00100000	les 32 octets suivants dans le sous-système TC
marque d'identification de transaction	01001001	= destination (Tableau 10/Q.773)
longueur	00000100	4 octets
valeur de l'identificateur de transaction	xxxxxxxx	comme dans BEGIN (message MRVT)
	xxxxxxxx	
	xxxxxxxx	
	xxxxxxxx	
marque de partie de composant	01101100	(Tableau 14/Q.773)
longueur	00011000	24 octets suivants
marque de type de composant	10100011	= retour d'erreur (Tableau 19/Q.773)
longueur	00010110	22 octets suivants
marque d'identificateur de composant	00000010	= id. d'invocation (Tableau 20/Q.773)
longueur	00000001	1 octet
valeur d'identificateur d'invocation	xxxxxxxx	comme dans le message MRVT (Corrélation)
marque de code d'erreur	00000010	Tableau 24/Q.773 (local)
longueur	00000001	1 octet
erreur de traitement	00001010	Figure 3/Q.754
marque de séquence de paramètres	00110000	= marque de séquence (Tableau 23/Q.773)
longueur	00001110	14 octets suivants
marque d'erreur spécifique	10100101	Figure 3/Q.754
longueur	00001100	12 octets suivants
marque de type d'erreur	10000000	Figure 3/Q.754
longueur	00000001	1 octet
faute	00000001	Figure 3/Q.754
paramètres d'erreur	10100001	Figure 3/Q.754
longueur	00000111	7 octets suivants
marque de type de défaillance	10000000	Figure 3/Q.754
longueur	00000010	2 octets
bits non utilisés	00000000	pas de bits
chaîne de défaillance	xxxxxxxx	dépend du type de défaillance
marque de trace d'émission	10000001	Figure 3/Q.754
longueur	00000001	1 octet
valeur de trace d'émission	0000000x	"Vrai" = 1, "Faux" = 0 Figure 3/Q.754)

Figure A.5/Q.754 – Exemple de message MRVA livré au sous-système SCCP (en cas d'erreur)

Nom du champ	Codage binaire	Référence/Commentaire
marque de type de message	01100010	= BEGIN (Tableau 8/Q.773)
longueur du message	00101100	44 octets suivants: sous-système TC
marque d'identification de transaction	01001000	= origine (Tableau 10/Q.773)
longueur	00000100	4 octets
valeur de l'identificateur de transaction	xxxxxxxx	sous-système TCAP
	xxxxxxxx	basé sur un dialogue
	xxxxxxxx	au niveau de l'utilisateur
	xxxxxxxx	
marque de partie de composant	01101100	(Tableau 14/Q.773)
longueur	00100100	36 octets suivants
marque de type de composant	10100001	= invocation (Tableau 19/Q.773)
longueur	00100010	34 octets suivants
marque d'identificateur de composant	00000010	= id. d'invocation (Tableau 20/Q.773)
longueur	00000001	1 octet
valeur d'identificateur d'invocation	xxxxxxxx	fourni par OMAP
marque de code opération	00000010	= local (Tableau 22/Q.773)
longueur	00000001	1 octet
code opération	00000000	= compte rendu d'événement (Figure 3/Q.754)
marque de séquence de paramètres	00110000	= marque de séquence (Tableau 23/Q.773)
longueur	00011010	26 octets suivants
marque de classe d'objets	10000000	(Figure 3/Q.754)
longueur	00000101	5 octets
valeur des tables d'acheminement MTP	00000000	Rec. UIT-T
	00010001	q
	10000101	85 => 754
	01110010	72 =>
	00000000	tables d'acheminement MTP 1992
marque d'instance d'objet	10000011	(Figure 3/Q.754)
longueur	00000010	2 octets
valeur d'instance d'objet	xxxxxxxx	point sémaphore de terminaison (OMAP)
	xxxxxxxx	<Destination testée>
marque de type d'événement	10000111	Figure 3/Q.754
longueur	00000001	1 octet
type d'événement	00000010	= trace d'itinéraire (Figure 3/Q.754)
marque de type d'information	10101000	Figure 3/Q.754
longueur	00001010	10 octets suivants
identificateur de réussite	10100000	Figure 3/Q.754
longueur	00001000	8 octets suivants
marque de code de point	00000100	= chaîne d'octets
longueur	00000010	2 octets
code de point	xxxxxxxx	
	xxxxxxxx	
marque de code de point	00000100	= chaîne d'octets
longueur	00000010	2 octets
code de point	xxxxxxxx	
	xxxxxxxx	

Figure A.6/Q.754 – Exemple de message MRVR livré au sous-système SCCP (en cas de réussite)

SERIES 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	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, de télégraphie, de télécopie, circuits téléphoniques 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 pour données et communication entre systèmes ouverts
Série Z	Langages de programmation