UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Q.754

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(03/93)

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»)

AVANT-PROPOS

L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'Union internationale des télécommunications (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 que les Commissions d'études de l'UIT-T doivent examiner et à propos desquels elles doivent émettre des Recommandations.

La Recommandation UIT-T Q.754, élaborée par la Commission d'études XI (1988-1993) de l'UIT-T, a été approuvée par la CMNT (Helsinki, 1-12 mars 1993).

NOTES

1 Suite au processus de réforme entrepris au sein de l'Union internationale des télécommunications (UIT), le CCITT n'existe plus depuis le 28 février 1993. Il est remplacé par le Secteur de la normalisation des télécommunications de l'UIT (UIT-T) créé le 1er mars 1993. De même, le CCIR et l'IFRB ont été remplacés par le Secteur des radiocommunications.

Afin de ne pas retarder la publication de la présente Recommandation, aucun changement n'a été apporté aux mentions contenant les sigles CCITT, CCIR et IFRB ou aux entités qui leur sont associées, comme «Assemblée plénière», «Secrétariat», etc. Les futures éditions de la présente Recommandation adopteront la terminologie appropriée reflétant la nouvelle structure de l'UIT.

2 Dans la présente Recommandation, le terme «Administration» désigne indifféremment une administration de télécommunication ou une exploitation reconnue.

TABLE DES MATIÈRES

Recommandation Q.754

DÉFINITIONS DES ÉLÉMENTS DE SERVICE D'APPLICATION POUR LA GESTION DU SYSTÈME DE SIGNALISATION N° 7

(Helsinki, 1993)

1 Introduction

En cas de divergence entre les Recommandations Q.753 et Q.754, c'est la première qui prévaut.

La présente Recommandation contient la définition des éléments de service d'application pour le sous-système pour l'exploitation, la maintenance et la gestion (OMASE) (*operation, maintenance and administration part application service element*). Ceux-ci fournissent les services appelés au moyen des primitives OM-EVENT-REPORT et OM-CONFIRMED-ACTION franchissant la frontière entre OMASE-User et OMASE (voir dans la Recommandation Q.753, le diagramme et les correspondances entre les services appelés dans OMASE-User et dans OMASE).

Les services OMASE sont dérivés de ceux définis dans le protocole service commun d'information de gestion (CMIP) (*common management information protocol*)[1)].

Les primitives OMASE sont définies à l'article 6; la syntaxe formelle définie à la Figure 3 utilise les macros OPERATION et ERROR du gestionnaire de transactions (TC) (*transaction capabilities*). L'interfonctionnement OMASE/TC figure également à l'article 6.

Les OMASE fournissent des opérations qui permettent à l'administration du réseau, via le processus de gestion OMAP et d'OMASE-User, d'effectuer des essais pour la vérification de l'acheminement dans le sous-système transport de messages (MRVT) (*MTP routing verification test*) et des essais pour la vérification de l'acheminement dans le SCCP (SRVT) (*SCCP routing verification test*) ainsi que des essais de validation de circuits (CVT) (*circuit validation tests*). La présente Recommandation contient la définition des ASE pour des essais MRVT, SRVT et CVT.

Les essais SRVT en question sont ceux destinés à l'essai spécifique du 3.2.2/Q.753.

Les arguments utilisés pour des primitives franchissant la frontière entre le processus de gestion OMAP et OMASE-User, pour les primitives franchissant les frontières OMASE-User/OMASE et OMASE/TC contiennent la même information s'ils ont le même nom. Ces arguments sont définis dans la présente Recommandation.

2 MTP

2.1 Essai pour la vérification de l'acheminement dans le MTP (MRVT)[2)]

L'essai MRVT lancé à l'origine de l'essai se traduit par une primitive OM-CONFIRMED-ACTION qui s'utilise de OMASE-User à OMASE et qui contient la commande testRoute en tant que paramètre. Si une trace des voies d'acheminement est requise ou s'il y a une anomalie, la primitive OM-EVENT-REPORT, qui contient l'événement routeTrace en tant que paramètre, est appelée à l'expéditeur de l'essai depuis OMASE.

L'essai testRoute est spécifié à l'aide du macro CNF-ACTION défini à la Figure 3; routeTrace est spécifié au moyen du macro EVENT défini à la Figure 3.

En ce qui concerne l'essai MRVT, ObjectClass indique les tables d'acheminement MTP, et ObjectInstance contient le code de point de la destination de l'essai. L'Action testRoute utilise le message BEGIN (MRVT) et le résultat (MRVA) est renvoyé dans un message END. L'événement routeTrace (MRVR) utilise un message BEGIN avec une terminaison prédéterminée.

2.1.1 testRoute Action (Action testRoute)

testRoute Action est lancée pour déclencher une procédure d'essai pour la vérification de l'acheminement dans le MTP. Au point initiateur, ce lancement est demandé par l'Administration via le [système d'information de gestion (MIS) (*management information system*)] MIS-User ou une interface locale, par le processus de gestion OMAP et

---

1) Le protocole CMIP est défini dans la norme ISO IS 9596 et dans la Recommandation X.711.

2) La description de la notation formelle est donnée dans les Recommandations X.208 et X.209.

OMASE-User. Dans les points suivants, l'action est implicitement demandée par la réception d'un appel de l'action testRoute. Une réponse satisfaisante indique la réussite complète de l'essai au point qui l'a lancé et, implicitement, à tous les points suivants où l'essai a été lancé. Une indication d'échec est renvoyée pour indiquer l'échec de l'essai en ce point ou dans un point suivant.

testRoute CNF_ACTION	*Temporisation = T1*	*Classe = 1*	*Code = 00000001*
ActionArg		*O/M*	*Référence*
initiatingSP		M	2.1.1.1.1
traceRequested		M	2.1.1.1.2
threshold		M	2.1.1.1.3
pointCodesTraversed		M	2.1.1.1.4
ActionResult			
vide			
Linked Operations			
N/A			
Specific Errors			*Référence*
failure			2.1.1.3.1
partialSuccess			2.1.1.3.2
O Optionnel M Obligatoire			

```
testRoute CNF-ACTION
ACTIONINFOARG SEQUENCE {
                              initiatingSP          [0] IMPLICIT PointCode,
                              traceRequested        [1 ] IMPLICIT BOOLEAN,
                              threshold             [2] IMPLICIT INTEGER,
                              pointCodesTraversed   [3] IMPLICIT PointCodeList
}
SPECIFICERRORS                {failure, partialSuccess}
::= 1
```

2.1.1.1 testRoute Action Arguments (Arguments de l'Action essayer-Acheminement)

2.1.1.1.1 initiatingSP (SP-Initiateur)

initiatingSP identifie l'origine initiatrice de l'essai. Il est de type PointCode, défini comme une chaîne d'octets.

Paramètre	*Code*
initiatingSP	10000000
Contenu	
Bit 0 contains the first bit of the Point Code, Bit 1 contains the second bit of the Point Code, etc.	

```
PointCode ::= OCTET STRING
```

2.1.1.1.2 traceRequested (traceDemandée)

traceRequested indique qu'une trace de toutes les routes utilisées pour atteindre la destination doit être donnée à l'initiateur (l'événement trace-Route est décrit en 2.1.2). Il est de type BOOLEAN.

Paramètre	*Code*
traceRequested	10000001
Contenu	*Signification*
TRUE (= 1)	La trace a été demandée, renvoyer l'information de trace en cas de succès ou d'échec
FALSE (= 0)	La trace n'a pas été demandée, renvoyer l'information de trace seulement en cas d'échec

2.1.1.1.3 threshold (seuil)

L'initiateur fixe un seuil maximal de points sémaphores (SP) (*signalling points*) qui peuvent être traversés au cours de l'essai (incluant l'initiateur si c'est un STP). Cela permet de détecter les routes trop longues. Ce seuil est un nombre entier de SP. Il est de type INTEGER.

Paramètre	*Code*
threshold	10000010
Contenu	
Nombre entier	

2.1.1.1.4 pointCodesTraversed (codesPointsTraversés)

Lorsque chaque SP intermédiaire est traversé, il ajoute son propre code de point à la liste des codes de points traversés. Cela permet de détecter les boucles; il s'agit également d'une information utile en cas d'échec ou lorsqu'une trace est demandée. C'est une liste de codes de points, elle est donc du type PointCodeList. pointCodeList pourrait être vide.

Paramètre	*Code*
pointCodesTraversed	10100011
Contenu	
Suite de codes de points, identifiés en tant que «PointCode» contenant le code de point exact	

```
PointCodeList ::= SEQUENCE OF PointCode
```

2.1.1.2 Action Results (Résultat Action)

Il n'y a pas de contenu dans une indication de succès retournée.

2.1.1.3 Action Errors (Erreurs Action)

Les SpecificErrors sont les erreurs possibles qui peuvent être détectées durant un essai. Elles sont propres à un essai. Ces erreurs spécifiques sont, de plus, les erreurs déjà identifiées dans le service om-Action-Confirmed et apparaissent en tant que paramètres dans Processing Failure Error (Erreur Echec Traitement).

2.1.1.3.1 failure (échec)

failure indique une condition d'échec total, où aucun acheminement n'est correct. Le plus souvent, cela sera utilisé en tant qu'indication d'échec en provenance du point qui détecte l'erreur et ne lance aucune autre testRoute Actions. SpecificError échec possède un paramètre indiquant le type d'erreur cause de l'échec de l'essai. Ce paramètre, failureType, est représenté comme une chaîne de bits. De plus, le second paramètre est utilisé lorsque failureType indique l'erreur UnknownInitiatingSP. traceSent indique si, oui ou non, un Evénement routeTrace a été lancé afin de donner l'information de trace. Cela est nécessaire car le point qui détecte l'erreur ne peut pas envoyer routeTrace, donc le point précédent doit l'envoyer, traceSent est de type BOOLEAN.

Erreur Spécifique	*Code*
failure	00000001
Paramètres	*Références*
failureType	2.1.1.3.1
traceSent	2.1.1.3.1

Paramètre	*Code*
failureType	10000000
Bit	*Signification*
0	detectedLoop
1	excessiveLengthRoute
2	unknownObjectInstance
3	routeInaccessible
4	processingFailure
5	unknownInitiatingSP
6	timerExpired
7	sPNotAnSTP

Paramètre	*Code*
traceSent	10000001
Contenu	*Signification*
TRUE	L'information de trace a été envoyée
FALSE	L'information de trace n'a pas été envoyée

```
failure         SPECIFIC-ERROR
                PARAMETER SEQUENCE      { failureType       [0] IMPLICIT FailureString,
                                          traceSent         [l] IMPLICIT BOOLEAN }
                ::= 1
```

```
FailureString ::= BIT STRING
                        {detectedLoop (0),
                        excessiveLengthRoute (1),
                        unknownObjectlnstance (2),
                        routeInaccessible (3),
                        processingFailure (4),
                        unknownInitiatingSP (5),
                        timerExpired (6),
                        sPNotAnSTP (7)}
```

2.1.1.3.2 partialSuccess (succès Partiel)

Cette indication est donnée lorsqu'au moins une invocation testRoute Cnf a échoué et qu'au moins un a réussi (au moins partiellement). Dans ce cas, chaque type d'erreur ayant été détecté sera noté et envoyé dans la réponse finale. Le format et le contenu d'un partial success sont les mêmes qu'un échec.

Erreur Spécifique	*Code*
partialSuccess	00000010
Paramètres	*Références*
failureType	2.1.1.3.1
traceSent	2.1.1.3.1

```
partialSuccess  SPECIFIC-ERROR
                PARAMETER SEQUENCE      { failureType       [0] IMPLICIT FailureString,
                                          traceSent         [1] IMPLICIT BOOLEAN }
                ::= 2
```

2.1.2 routeTrace Event (Evénement traceRoute)

routeTrace Event rend compte de l'information de trace. L'information de trace comprend zéro, un ou plusieurs codes de point, tels que le code du point détectant une erreur ou la liste complète des codes des points traversés dans une route. Cet événement est lancé soit sur une demande explicite du point origine (indiquée par traceRequested, voir 2.1.1.1.2) soit en cas d'échec dans n'importe quel point de la route. Cet événement n'est pas confirmé et par conséquent aucune réponse n'est attendue (pas d'indication d'erreur ou de succès).

routeTrace EVENT	Temporisation = 0	Classe = 4	Code = 00000010
EventInfo		*O/M* (Note)	*Référence*
success		O	2.1.2.1.1
detectedLoop		O	2.1.2.1.2
excessiveLengthRoute		O	2.1.2.1.3
unknownObjectInstance		O	2.1.2.1.4
routeInaccessible		O	2.1.2.1.5
processingFailure		O	2.1.2.1.6
unknownInitiatingSP		O	2.1.2.1.7
timerExpired		O	2.1.2.1.8
sPNotAnSTP		O	2.1.2.1.9
NOTE – Un et un seul de ces paramètres doit être présent.			

```
routeTrace          EVENT
                    EVENTINFO CHOICE {
                                        success                 [0] IMPLICIT PointCodeList,
                                        detectedLoop            [1] IMPLICIT PointCodeList,
                                        excessiveLengthRoute    [2] IMPLICIT PointCodeList,
                                        unknownObjectInstance   [3] IMPLICIT NULL,
                                        routeInaccessible       [4] IMPLICIT PointCode,
                                        processingFailure       [5] IMPLICIT NULL,
                                        unknownInitiatingSP     [6] IMPLICIT PointCode,
                                        timerExpired            [7] IMPLICIT PointCodeList,
                                        sPNotAnSTP              [8] IMPLICIT PointCodeList,
                    }
                    ::= 2
```

2.1.2.1 Event Information (Info Evénement)

2.1.2.1.1 success (succès)

Comprend, en cas de succès complet, la trace des codes de point (un ou plus) des SP traversés.

Paramètre	*Code*
success	10100000
Contenu	*Références*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	2.1.1.1.4

2.1.2.1.2 detectedLoop (boucle Détectée)

Comprend, lorsqu'une boucle est détectée, les codes des points (trois ou plus) compris dans la boucle.

Paramètre	*Code*
detectedLoop	10100001
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	2.1.1.1.4

2.1.2.1.3 excessiveLengthRoute (route Trop Longue)

Comprend la route concernée, lorsqu'une route trop longue est détectée (seuil dépassé).

Paramètre	*Code*
excessiveLengthRoute	10100010
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	2.1.1.1.4

2.1.2.1.4 unknownObjectInstance (instance Objet Inconnue)

Si l'instance d'objet est inconnue, aucune information supplémentaire n'est nécessaire.

Paramètre	*Code*
unknownObjectInstance	10000011
Contenu	*Référence*
vide	–

2.1.2.1.5 routeInaccessible (route Indisponible)

Comprend le code du point où une route a été trouvée indisponible.

Paramètre	*Code*
routeInaccessible	10000100
Contenu	*Référence*
Le bit 0 contient le premier bit du code de point, Le bit 1 contient le second bit du code de point, etc.	2.1.1.1.1

2.1.2.1.6 processingFailure (échec Traitement)

Si un échec de traitement intervient, aucune information supplémentaire n'est requise.

Paramètre	*Code*
processingFailure	10000101
Contenu	*Référence*
vide	–

2.1.2.1.7 unknownInitiatingSP (SP Initiateur Inconnu)

Comprend le code du point détectant le SP initiateur Inconnu.

Paramètre	*Code*
unknownInitiatingSP	10000110
Contenu	*Référence*
Le bit 0 contient le premier bit du code de point, Le bit 1 contient le second bit du code de point, etc.	2.1.1.1.1

2.1.2.1.8 timerExpired (chute Temporisation)

Contient le (les) Code(s) de Point(s) du (des) point(s) pour lequel (lesquels) aucun résultat à une testRoute Action n'a été reçu.

Paramètre	*Code*
timerExpired	10100111
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» contenant le code de point exact	2.1.1.1.4

2.1.2.1.9 SPNotANSTP (SP Autre que STP)

Comprend, si le SP intermédiaire qui reçoit un message MRVT n'est pas doté de la fonction de transfert du MTP, la liste des SP traversés pour atteindre ce SP.

Paramètre	*Code*
sPNotAnSTP	10101000
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» contenant le code de point exact	2.1.1.1.4

3 Sous-système commande des connexions sémaphores (SCCP)

3.1 ASE d'essai pour la vérification de l'acheminement dans le SCCP (SRVT)

Les fonctions spécifiques du SRVT sont définies en 3.2.2/Q.753. L'essai SRVT lancé à l'origine de l'essai se traduit par une primitive OM-CONFIRMED-ACTION qui s'utilise de OMASE-User à OMASE et qui contient la commande testRoute en tant que paramètre. Si une trace des voies d'acheminement est requise ou s'il y a une anomalie, la primitive OM-EVENT-REPORT, qui contient l'événement routeTrace en tant que paramètre, est appelée à l'expéditeur de l'essai depuis OMASE.

L'essai testRoute est spécifié à l'aide du macro CNF-ACTION défini à la Figure 3, routeTrace est spécifié au moyen du macro EVENT défini à la Figure 3.

La ObjectClass indique les tables utilisées pour la traduction d'appellation globale SCCP et ObjectInstance contient l'indicateur d'appellation globale et l'appellation globale testée. Le codage de l'indicateur d'appellation globale (GTI) est défini dans l'indicateur d'adresse du SCCP. L'action testRoute (SRVT) utilise le message BEGIN et le résultat (SRVA) est renvoyé dans un message END. L'Evénement routeTrace (SRVR) utilise un message BEGIN avec une terminaison prédéterminée.

3.1.1 testRoute Action (Action testRoute)

La testRoute Action est lancée pour déclencher une procédure d'essai pour la vérification de l'acheminement dans le SCCP. Au point initiateur, ce lancement est demandé par l'Administration via le MIS-User ou une interface locale, par le processus de gestion OMAP et OMASE-User. Dans les points suivants, l'Action est implicitement demandée par la réception d'une invocation testRoute Action. Une réponse de succès indique la réussite complète de l'essai au point qui l'a lancé et, implicitement, à tous les points suivants où l'essai a été lancé. Une indication d'échec est renvoyée pour indiquer l'échec de l'essai en ce point ou dans un point suivant.

testRoute CNF-ACTION	*Temporisation* = T2	*Classe* = 1	*Code* = 00000001
ActionArg		*O/M*	*Références*
initiatingSP		M	3.1.1.1.1
traceRequested		M	3.1.1.1.2
threshold		M	3.1.1.1.3
pointCodesTraversed		M	3.1.1.1.4
formIndicator		M	3.1.1.1.5
mtpBackwardRoutingRequested		M	3.1.1.1.6
testInitiatorGT		O	3.1.1.1.7
destinationPC		O	3.1.1.1.8
destinationSSN		O	3.1.1.1.9
backupDPC		O	3.1.1.1.10
backupSSN		O	3.1.1.1.11
originalGT		O	3.1.1.1.12
ActionResult			
vide			
Linked Operations			
N/A			
Specific Errors			*Références*
failure			3.1.1.3.1
partialSuccess			3.1.1.3.2

```
testRoute CNF-ACTION
ACTIONINFOARG SEQUENCE {
        initiatingSP                    [0] IMPLICIT PointCode,
        traceRequested                  [1] IMPLICIT BOOLEAN,
        threshold                       [2] IMPLICIT INTEGER,
        pointCodesTraversed             [3] IMPLICIT PointCodeList,
        formIndicator                   [4] IMPLICIT FormIndicator,
        mtpBackwardRoutingRequested     [5] IMPLICIT BOOLEAN,
        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
}
SPECIFICERRORS                          { failure, partialSuccess }
::= 1
```

3.1.1.1 testRoute Action Arguments (Arguments de l'Action essayerAcheminement)

3.1.1.1.1 initiatingSP (SP Initiateur)

Le initiatingSP identifie l'origine initiatrice de l'essai. Il est de type PointCode.

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

```
PointCode ::= OCTET STRING
```

3.1.1.1.2 traceRequested (trace Demandée)

traceRequested indique qu'une trace de toutes les routes utilisées pour atteindre la destination doit être donnée à l'initiateur (routeTrace Event est décrit en 3.1.2). Il est de type BOOLEAN.

Paramètre	*Code*
traceRequested	10000001
Contenu	*Signification*
TRUE (= 1)	La trace a été demandée, renvoyer l'information de trace en cas de succès ou d'échec
FALSE (= 0)	La trace n'a pas été demandée, renvoyer l'information de trace seulement en cas d'échec

3.1.1.1.3 treshold (seuil)

L'initiateur fixe un seuil maximal de points sémaphores traducteurs (TSP) (*translation signalling points*) qui peuvent être traversés au cours de l'essai (incluant l'initiateur si c'est un nœud relais SCCP). Cela permet de détecter les routes trop longues. Ce seuil est un nombre entier de SP. Il est donc de type INTEGER.

Paramètre	*Code*
threshold	10000010
Contenu	
Nombre entier	

3.1.1.1.4 pointCodesTraversed (codesPointsTraversés)

Lorsqu'un SP traducteur est traversé, il ajoute son propre code de point à la liste des codes de points traversés. Cela permet de détecter les boucles; il s'agit également d'une information utile en cas d'échec ou lorsqu'une trace est demandée. C'est une liste de codes de points, elle est donc du type PointCodeList. PointCodeList pourrait être vide.

Paramètre	*Code*
pointCodesTraversed	10100011
Contenu	
Suite de codes de point identifiée en tant que «PointCode» et contenant le code de point exact	

```
PointCodeList ::= SEQUENCE OF PointCode
```

3.1.1.1.5 formIndicator (indicateur Type)

formIndicator identifie le type du message SRVT (c'est-à-dire Demande, Vérification ou Comparaison). Il est de type INTEGER et a les valeurs indiquées ci-dessous.

Paramètre	*Code*
formIndicator	10000100
Contenu	
Valeur 0 = Comparaison Valeur 1 = Pas de Comparaison	

```
FormIndicator ::=        INTEGER
                           { compare (0),
                           noCompare (l) }
```

3.1.1.1.6 mtpBackwardRoutingRequested (acheminement Vers Arrière MTP Demandé)

L'argument mtpBackwardRoutingRequested détermine si l'acheminement vers l'arrière du MTP à destination de l'OPC est nécessaire pour que l'essai réussisse. Il est de type BOOLEAN.

Paramètre	*Code*
mtpBackwardRoutingRequested	10000101
Contenu	
TRUE (= 1) Acheminement demandé FALSE (= 0) Acheminement non demandé	

3.1.1.1.7 testInitiatorGT (AG Initiateur Essai)

testInitiatorGT identifie l'indicateur d'appellation globale et l'appellation globale de l'initiateur. Il est de type OCTET STRING.

Paramètre	*Code*
testInitiatorGT	10000110
Contenu	
Octet 1 = Indicateur d'appellation globale Octet 2, 3, ... = Appellation globale de l'Initiateur	

GlobalTitle ::= OCTET STRING

3.1.1.1.8 destinationPC (code Point Destination)

destinationPC identifie le code de point de destination (point sémaphore de destination de premier choix (PPC) ou Code de point traducteur (TPC). Il est de type PointCode.

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

3.1.1.1.9 destinationSSN (SSN Destination)

destinationSSN identifie le numéro du sous-système de destination. Il est du type OCTET STRING.

Paramètre	*Code*
destinationSSN	10001000
Contenu	
Le bit 0 contient le premier bit du code de point, Le bit 1 contient le second bit du code de point, etc.	

SubsystemNumber ::= OCTET STRING

3.1.1.1.10 backupDPC (DPC Additionnel)

backupDPC identifie le code de point de destination (SPC) additionnel. Il est de type PointCode.

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

3.1.1.1.11 backupSSN (SSN Additionnel)

backupSSN identifie le numéro du sous-système de destination additionnel. Il est de type OCTET STRING.

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

3.1.1.1.12 originalGT

Le champ originalGT n'est présent dans un message SRVT que si la traduction d'un GT dans l'adresse du demandé donne, ou a déjà donné, un GT de remplacement.

Dans ce cas, le champ dans un message SRVT qu'il convient d'envoyer se présente de la manière suivante:

i) si le message SRVT à expédier n'est pas la formule de comparaison, et que l'essai est lancé par la réception d'un message SRVT contenant un originalGT, le champ est copié de l'autre côté, ou

ii) dans tous les autres cas, l'originalGT expédié est le GT de l'adresse du demandé pour le message SRVT avant traduction.

Ce champ est utilisé comme titre GT de l'adresse du demandé et est inséré dans tout message de test SRVR envoyé. Pour la forme comparative du message SRVT, ce champ est utilisé par le point TSP qui le reçoit afin de vérifier que sa conversion fournit bien le titre global dans le champ d'adresse du demandé inséré dans le message de test SRVT comparatif reçu.

Le type du paramètre originalGT est GlobalTitle.

Paramètre	*Code*
originalGT	10001011
Contenu	
Octet 1 = Indicateur de titre global Octet 2, 3,... = Titre global original	

3.1.1.2 Action Results (Résultat Action)

Il n'y a pas de contenu dans une indication de succès retournée.

3.1.1.3 Action Errors (Erreurs Action)

Les specificErrors sont les erreurs possibles qui peuvent être détectées durant un essai. Elles sont propres à un essai. Ces erreurs spécifiques sont, de plus, les erreurs déjà identifiées dans le service om-Action-Confirmed et apparaissent en tant que paramètres dans l'erreur d'échec de traitement.

3.1.1.3.1 failure (échec)

failure indique une condition d'échec où une traduction n'a pas abouti ou était incorrecte. Le plus souvent, cela sera utilisé en tant qu'indication d'échec en provenance du point qui détecte l'erreur et ne lance aucune autre testRoute Actions. failure SpecificError possède un paramètre indiquant le type d'erreur cause de l'échec de l'essai. Ce paramètre, failureType, est représenté par une chaîne de bits. De plus, le second paramètre doit être utilisé lorsque failureType indique l'erreur unknowInitiatingSP. traceSent indique si, oui ou non, un routeTrace Event a été invoqué afin de donner l'information de trace. Cela est nécessaire car le point qui détecte l'erreur ne peut pas envoyer routeTrace, donc le point précédent doit l'envoyer. traceSent est de type BOOLEAN et son utilisation est facultative.

ErreurSpécifique	*Code*
failure	00000001
Paramètre	*Références*
failureType	3.1.1.3.1
traceSent	3.1.1.3.1

Paramètre	*Code*
failureType	10000000
Bit	*Signification*
0	detectedLoop
1	excessiveLengthRoute
2	unknownObjectInstance
3	routeInaccessible
4	processingFailure
5	unknownInitiatingSP
6	timerExpired
7	wrongSP
8	incorrectTranslation-Primary
9	incorrectTranslation-Secondary
10	incorrectTranslation-Intermediate
11	notPrimaryDestination
12	notSecondaryDestination
13	notRecognizedPrimary
14	notRecognizedSecondary
15	routingProblem

Paramètre	*Code*
traceSent	10000001
Contenu	*Signification*
TRUE	L'information de trace a été envoyée
FALSE	L'information de trace n'a pas été envoyée

```
failure         SPECIFIC-ERROR
                PARAMETER SEQUENCE      { failureType       [0] IMPLICIT FailureString,
                                          traceSent         [1 ] IMPLICIT BOOLEAN }
                ::= 1
```

```
FailureString ::= BIT STRING
                  { detectedLoop (0),
                  excessiveLengthRoute (1),
                  unknownObject (2),
                  routeInaccessible (3),
                  processingFailure (4),
                  unknownInitiatingSP (5),
                  timerExpired (6),
                  wrongSP (7),
                  incorrectTranslation-Primary (8),
                  incorrectTranslation-Secondary (9),
                  incorrectTranslation-Intermediate (10),
                  notPrimaryDestination (11),
                  notSecondaryDestination (12),
                  notRecognizedPrimary (13),
                  notRecognizedSecondary (14),
                  routingProblem (15) }
```

3.1.1.3.2 partialSuccess (succèsPartiel)

Cette indication est donnée lorsqu'au moins une invocation testRoute Cnf Action a échoué et qu'au moins un a réussi (au moins partiellement). Dans ce cas, chaque type d'erreur ayant été détecté sera noté et envoyé dans la réponse finale. Le format et le contenu d'un partial success sont les mêmes que ceux d'un échec.

ErreurSpécifique	*Code*
partialSuccess	00000010
Paramètres	*Références*
failureType	3.1.1.3.1
traceSent	3.1.1.3.1

```
partialSuccess    SPECIFIC-ERROR
                  PARAMETER SEQUENCE      { failureType    [0] IMPLICIT FailureString,
                                            traceSent      [1] IMPLICIT BOOLEAN }
                  ::= 2
```

3.1.2 routeTrace Event (Evénement trace-Route)

routeTrace Event rend compte de l'information de trace. L'information de trace comprend un ou plusieurs codes de point, tels que la liste complète des codes de points traducteurs traversés dans une route. Cet événement est invoqué soit sur une demande explicite du point origine (indiquée par traceRequested, voir 3.1.1.1.2), soit en cas d'échec dans n'importe quel point de la route. Cet événement n'est pas confirmé et par conséquent aucune réponse n'est attendue (pas d'indication d'erreur ou de succès).

routeTrace EVENT	*Temporisation* = 0	*Classe* = 4	Code = 00000010
EventInfo		*O/M* (Note)	*Références*
success		O	3.1.2.1.1
detectedLoop		O	3.1.2.1.2
excessiveLengthRoute		O	3.1.2.1.3
unknownObjectInstance		O	3.1.2.1.4
routeInaccessible		O	3.1.2.1.5
processingFailure		O	3.1.2.1.6
unknownInitiatingSP		O	3.1.2.1.7
timerExpired		O	3.1.2.1.8
wrongSP		O	3.1.2.1.9
incorrectTranslation-Primary		O	3.1.2.1.10
incorrectTranslation-Secondary		O	3.1.2.1.11
incorrectTranslation-Intermediate		O	3.1.2.1.12
notPrimaryDestination		O	3.1.2.1.13
notSecondaryDestination		O	3.1.2.1.14
notRecognizedPrimary		O	3.1.2.1.15
notRecognizedSecondary		O	3.1.2.1.16
routingProblem		O	3.1.2.1.17
NOTE – Une et une seule de ces indications doit être présente.			

```
routeTrace      EVENT
                EVENTINFO CHOICE {
                success                          [0] IMPLICIT PointCodeList,
                detectedLoop                     [1] IMPLICIT PointCodeList,
                excessiveLengthRoute             [2] IMPLICIT PointCodeList,
                unknownObjectInstance            [3] IMPLICIT NULL,
                routeInaccessible                [4] IMPLICIT PointCode,
                processingFailure                [5] IMPLICIT NULL,
                unknownInitiatingSP              [6] IMPLICIT PointCode,
                timerExpired                     [7] IMPLICIT PointCodeList,
                wrongSP                          [8] IMPLICIT PointCodeList,
                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 }
                ::= 2
```

3.1.2.1 Event Information (Info Evénement)

3.1.2.1.1 success (succès)

Comprend, en cas de succès complet, la trace des codes de point (un ou plus) des nœuds relais SCCP traversés.

Paramètre	*Code*
success	10100000
Contenu	*Référence*
Suite d'un ou de plusieurs codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.2 detectedLoop (boucle Détectée)

Comprend, lorsqu'une boucle est détectée, les codes des points (trois ou plus) compris dans la boucle.

Paramètre	*Code*
detectedLoop	10100001
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.3 excessiveLengthRoute (route Trop Longue)

Comprend la route concernée, lorsqu'une route trop longue est détectée (seuil dépassé).

Paramètre	*Code*
excessiveLengthRoute	10100010
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.4 unknownObjectInstance (instance Objet Inconnue)

Si l'objet est inconnu, aucune information supplémentaire n'est nécessaire. Pour le SRVT, cela correspond au cas où il n'existe pas de données de traduction pour la GTI + GT.

Paramètre	*Code*
unknownObjectInstance	10000011
Contenu	*Référence*
vide	–

3.1.2.1.5 routeInaccessible (route Indisponible)

Comprend le code du point où une route a été trouvée indisponible.

Paramètre	*Code*
routeInaccessible	10000100
Contenu	*Référence*
Le bit 0 contient le premier bit du code de point, Le bit 1 contient le second bit du code de point, etc.	3.1.1.1.1

3.1.2.1.6 processingFailure (échec Traitement)

Si un échec de traitement intervient, aucune information supplémentaire n'est requise.

Paramètre	*Code*
processingFailure	10000101
Contenu	*Référence*
vide	–

3.1.2.1.7 unknownInitiatingSP (SP Initiateur Inconnu)

Comprend le code du point détectant le point sémaphore initiateur inconnu.

Paramètre	*Code*
unknownInitiatingSP	10000110
Contenu	*Référence*
Le bit 0 contient le premier bit du code de point, Le bit 1 contient le second bit du code de point, etc.	3.1.1.1.1

3.1.2.1.8 timerExpired (chute Temporisation)

Contient le(s) code(s) de point(s) du (des) point(s) pour lequel (lesquels) aucun résultat à une testRoute Action n'a été reçu.

Paramètre	*Code*
timerExpired	10100111
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.9 wrongSP (SP Erroné)

Comprend la liste de tous les SP traducteurs traversés dans une route à destination du SP non valide.

Paramètre	*Code*
wrongSP	10101000
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.10 incorrectTranslation – Primary (traduction Incorrecte destination de premier choix)

Comprend la liste de tous les SP traducteurs traversés dans la route vers la destination de premier choix incorrecte.

Paramètre	*Code*
incorrectTranslation-Primary	10101001
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.11 incorrectTranslation – Secondary (traduction Incorrecte destination de second choix)

Comprend la liste de tous les SP traducteurs traversés dans la route vers la destination de second choix incorrecte.

Paramètre	*Code*
incorrectTranslation-Secondary	10101010
Contenu	*Référence*
Sequence of Point Codes,	3.1.1.1.4
Suite de code de point identifiée comme «PointCode» et contenant le code de point exact	

3.1.2.1.12 incorrectTranslation – Intermediate (traduction Intermédiaire Incorrecte)

Comprend la liste de tous les SP traducteurs traversés dans la route vers le point intermédiaire incorrect.

Paramètre	*Code*
incorrectTranslation-Intermediate	10101011
Contenu	*Référence*
Sequence of Point Codes, Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.13 notPrimaryDestination (destination Autre Que Premier Choix)

Comprend la liste de tous les SP traducteurs traversés dans la route vers la destination de premier choix non valide.

Paramètre	*Code*
notPrimaryDestination	10101100
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.14 notSecondaryDestination (destination Autre Que Second Choix)

Comprend la liste de tous les SP traducteurs traversés dans la route vers la destination de second choix non valide.

Paramètre	*Code*
notSecondaryDestination	10101101
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.15 notRecognizedPrimary (destination Premier Choix Non Reconnue)

Contient la liste de tous les SP traducteurs traversés dans la route vers la destination de second choix.

Paramètre	*Code*
notRecognizedPrimary	10101110
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.16 notRecognizedSecondary (destination Second Choix Non Reconnue)

Comprend la liste de tous les SP traducteurs traversés dans la route vers la destination de premier choix.

Paramètre	*Code*
notRecognizedSecondary	10101111
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

3.1.2.1.17 routingProblem (problèmeAcheminement)

Comprend la liste de tous les SP traducteurs traversés dans la route à destination du problème d'acheminement éventuel. Ce problème survient lorsque le code de point issu de la traduction n'est pas reconnu.

Paramètre	*Code*
routingProblem	10110000
Contenu	*Référence*
Suite de codes de point identifiée comme «PointCode» et contenant le code de point exact	3.1.1.1.4

4 Gestion des circuits

4.1 Elément de service d'application pour les essais de validation d'un circuit (CVT)

L'élément de service d'application pour les essais de validation de circuit fournit les services accessibles via la primitive om-Action-Confirmed décrite à la Figure 3. Il utilise une instance fondée sur une des classes d'objet de gestion de circuits définies dans la Recommandation Q.751. BaseManagedObjectClass indique cvt-Cic-Tables-1992, et BaseManagedObjectInstance identifie le cktGrpInfo (information de groupe de circuits identifiant l'identificateur préalablement défini pour le circuit et son groupe, convenue entre les commutateurs aux extrémités des groupes de circuits) et son CIC connu dans le SP qui envoie.

4.1.1 cktValidTestCnf Action (Action confirmée essayer validation Ckt)

La demande d'essai de validation d'un circuit et l'envoi ultérieur de la réponse de validation du circuit sont mis en correspondance avec une action confirmée. L'action à entreprendre est la demande d'essai à l'extrémité distante.

cktValidTest CNF-ACTION	*Temporisation* = T_c	*Classe* = 1	*Code* = 00000001
ActionArg		O/M	*Références*
requestingSP		M	4.1.1
timer		O	4.1.1
ActionResult			*Référence*
success			4.1.1
Linked Operations			
N/A			
Specific Errors			
failure			

Les valeurs possibles de T_c et de la temporisation sont données en 4.2/Q.753.

```
cktValidTest                CNF-ACTION
ACTIONARG SEQUENCE {
                            requestingSP        RequestingSP,
                            timer               Timer OPTIONAL }
                            ACTIONRESULT                        Success
                            SPECIFICERRORS                      { failure }
                            ::= 3
```

4.1.2 Action Arguments (Arguments Action)

Le SP initiateur est le code de point du point sémaphore qui engage la procédure d'essai. Il est de type chaîne d'octets, comme indiqué ci-dessous.

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

4.1.3 Action Results (Résultats Action)

En cas de succès, les Action Results sont renvoyés dans un composant résultat. Le contenu des deux paramètres doit être défini d'après la procédure CVT.

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

A noter que CktGrpInfo est défini comme OCTET STRING.

4.1.4 Specific Error (Erreur Spécifique)

L'argument Specific Error indique l'échec et la raison de l'échec. Le contenu des deux paramètres doit être défini à l'aide de la procédure CVT.

```
failure     SPECIFIC-ERROR
            PARAMETER SEQUENCE {
                cktGrpInfo [0] IMPLICIT CktGrpInfo,
                cICName    [1] IMPLICIT OCTET STRING OPTIONAL
            }
```

A noter que CktGrpInfo est défini comme OCTET STRING.

Il existe diverses causes échec des essais de validation d'un circuit, à savoir:

a) CIC non affecté à l'extrémité rapprochée;

b) données de l'extrémité rapprochée inadaptées au circuit;

c) non réception d'une tonalité valable à l'extrémité rapprochée;

d) expiration de la temporisation d'essai globale T_c avant réception du message CVR;

e) réception du message CVR avant l'obtention de la synchronisation pendant l'essai de configuration binaire;

f) expiration de T_c avant l'obtention de la synchronisation pendant l'essai de configuration binaire;

g) configuration binaire encore en cours de réception à l'expiration de T_c;

h) configuration binaire encore en cours de réception au moment où le message CVR est reçu;

i) réception de la tonalité à l'expiration de T_c;

j) réception de la tonalité au moment où le message CVR est reçu;

k) le CIC de l'extrémité rapprochée et le CIC de l'extrémité éloignée ne correspondent pas (contrôle de l'extrémité rapprochée au reçu du message CVR)

l) réception du message CVR indiquant la cause de l'échec:

- CIC non affecté à l'extrémité éloignée;
- données de l'extrémité éloignée inadaptées au circuit;
- caractéristiques du faisceau de circuits impossibles à obtenir à l'extrémité éloignée;

m) cause de l'échec non précisée.

5 Gestionnaire de transactions

Doit faire l'objet d'un complément d'étude.

6 Définitions générales

6.1 Objets et opérations

L'OMAP effectue des essais sur les objets tels que les tables utilisées pour l'acheminement dans le MTP et le SCCP. Ces objets sont décrits ici en tant que «Classes d'objets» et sont identifiés par un identificateur d'objet spécifié par le numéro de la présente Recommandation et le type d'objet. Cette structure est décrite pour les identificateurs d'objet de l'OMAP: mtp-Routing-Tables, sccp-Routing-Tables et cvt-Cic-Tables.

oMAP	OBJECT IDENTIFIER ::= { ccitt recommendation q754 }
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 classe d'objets des tables utilisées pour l'acheminement dans le MTP est 0011857200 (hexadécimal); pour les tables de l'acheminement dans le SCCP elle est 0011857201 (hexadécimal) et pour les tables CIC de CVT elle est 0011857205 (hexadécimal). Voir l'Annexe C/X.208 et 22/X.209.

Les Tableaux 1 et 2 montrent les primitives OM, la Figure 1 représente les opérations OMAP dérivés du CMIP (IS 9596) et la Figure 3 contient une syntaxe formelle des OMASE.

Les opérations dérivées du CMIP sont énumérées ci-dessous.

Les opérations actuellement définies sont	
0	eventReport
7	confirmedAction

6.2 Primitives et procédures du protocole OMASE

6.2.1 Considérations générales

Le protocole OMASE utilise le service TC tel qu'il est défini dans la Recommandation Q.771. Les paramètres Invoke ID et Dialogue ID correspondent à ceux définis pour le service TC.

L'OMASE est modélisé au moyen d'une Protocol Machine (désignée ci-après OMPM). L'abréviation APDU représente l'unité de données du protocole d'application; elle se réfère au contenu de la ou des primitives échangées entre OMASE et TC.

La Figure A.1 représente le modèle comprenant les TC et SCCP; l'OMPM réside dans l'OMASE. La Figure A.2 représente un exemple de cas particulier des primitives dans un essai MRV (mais sans OM-EVENT-REPORT).

6.2.2 OM-EVENT-REPORT

6.2.2.1 Primitive de service

La primitive OM-EVENT-REPORT utilisée entre OMASE-User et OMASE est définie dans le Tableau 1.

L'événement spécifique qui est survenu est interprété dans le contexte de la classe d'objet spécifiée.

TABLEAU 1/Q.754

Paramètres de la primitive OM-EVENT-REPORT

Nom de paramètre	Dem./Ind.
CallingPartyAddress	M
CalledPartyAddress	M
DialogueID	M
InvokeID	M
ManagedObjectClass	M
ManagedObjectInstance	M
EventType	M
EventTime	O
EventInfo	O

Définitions des paramètres

CallingPartyAddress: tel que défini dans l'adresse de l'entité demandeur du 2.2/Q.711.

CalledPartyAddress: tel que défini dans l'adresse de l'entité demandeur du 2.2/Q.711. Les adresses ci-dessus servent à identifier l'OMAP, respectivement aux SP demandeur et demandé: Pour MRVT, tous deux peuvent prendre la forme d'un code de point suivi d'un numéro de sous-système (OMAP); pour SRVT, ils ont une forme convenant au type d'acheminement SCCP appliqué à l'essai.

DialogueID: tel que défini dans les Recommandations Q.771-Q.775. Correspond à la Transaction Id. définie dans la Recommandation Q.772.

InvokeID: tel que défini dans la Recommandation Q.772.

ManagedObjectClass: identifie la classe des objets pour lesquels cet événement est signalé.

ManagedObjectInstance: identifie l'instance d'objet sur laquelle l'événement est signalé.

EventType: spécifie l'événement particulier que signale l'instance objet.

EventTime: spécifie l'heure à laquelle l'événement a été généré.

EventInfo: fournit des indications supplémentaires spécifiques sur l'événement.

6.2.2.2 Procédure de signalisation d'événement

6.2.2.2.1 Réception de demande OM-EVENT-REPORT

La procédure de signalisation d'événement est lancée à la réception de la primitive de demande OM-EVENT-REPORT. Quand cela arrive, l'OMPM construit l'APDU qui demande l'opération eventReport et transmet l'APDU au moyen du service TC-INVOKE et TC-BEGIN.

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

- Dialogue ID – Défini par OMASE-User.
- Invoke ID – Défini par OMASE-User.
- Operation – Mis à eventReport.
- Class – Mis à 4.
- Paramètres – Ceux qui suivent le mot «PARAMETER» dans la définition d'eventReport. La valeur du paramètre eventType spécifie l'action à entreprendre – elle doit indiquer routeTrace pour les procédures présentement définies.
- Timeout – Mis à 0 pour les essais MRVT et SRVT.

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

- Adresse de destination – Telle que reçue dans CalledPartyAddress de la primitive de demande OM-EVENT-REPORT.
- Adresse de départ – Telle que reçue dans CallingPartyAddress de la primitive de demande OM-EVENT-REPORT.
- Dialogue ID – Comme dans TC-INVOKE.

La primitive de demande N-UNITDATA finalement envoyée au SCCP suite à la réception de ces primitives TC request doit avoir une valeur du paramètre Sequence control qui correspond à «sequence garantie», alors que le paramètre Return option doit correspondre à «refuser le message en cas d'erreur». Voir 2.2.2/Q.711.

Après la transmission de l'APDU, l'OMPM met fin au dialogue en émettant la primitive de demande TC-END avec les paramètres Dialogue ID et Termination, ce dernier correspondant à «terminaison prédéterminée».

6.2.2.2.2 Réception de TC-BEGIN avec indication TC-INVOKE

Au reçu d'une APDU convenablement configurée qui demande l'opération eventReport à partir des primitives d'indication TC-BEGIN et TC-INVOKE, l'OMPM envoie une primitive d'indication OM-EVENT-REPORT. Si l'APDU n'est pas convenablement configurée, l'OMPM le refuse.

L'OMPM met fin au dialogue en émettant une primitive de demande TC-END avec les paramètres Dialogue ID et Termination, ce dernier correspondant à «terminaison prédéterminée».

6.2.2.2.3 Réception de TC-BEGIN avec indication TC-L-REJECT

Dans ce cas, l'OMPM émet une primitive de demande TC-END avec les paramètres Dialogue ID et Termination, ce dernier correspondant à «terminaison prédéterminée».

6.2.2.2.4 Réception de l'indication TC-P-ABORT

Dans ce cas, l'OMPM ignore l'indication TC-P-ABORT.

6.2.3 OM-CONFIRMED-ACTION

6.2.3.1 Primitive de service

Le service OM-CONFIRMED-ACTION est représenté dans le Tableau 2. L'action spécifique à effectuer est interprétée dans le contexte de la classe d'objet spécifiée. Ce service est un service confirmé (la réussite ou la défaillance est toujours signalée).

TABLEAU 2/Q.754

Service OM-CONFIRMED-ACTION

Nom du paramètre	Dem/Ind	Rép/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 Return Result (peut être vide).

b) Obligatoire dans le composant Return Error.

c) Ce paramètre figure uniquement dans la primitive de demande.

Définitions des paramètres

CallingPartyAddress: voir le Tableau 1.

CalledPartyAddress: voir le Tableau 1.

DialogueID: mis en correspondance par TCAP dans transaction ID, tel que défini dans la Recommandation Q.772.

InvokeID: tel que défini dans la Recommandation Q.772.

AccessControl: information à utiliser comme entrée pour accéder aux fonctions de commande.

BaseManagedObjectClass: identifie la classe des objets pour lesquels cette action est définie.

BaseManagedObjectInstance: identifie l'instance d'objet sur laquelle l'action doit avoir lieu.

ActionInfo: une séquence de ActionType et (facultativement) ActionInfoArg. ActionType est défini par le macro CNF-ACTION et spécifie une action particulière qu'il y a lieu d'exécuter sur l'instance d'objet. ActionInfoArg contient les paramètres relatifs à l'action qu'il convient d'effectuer.

ActionResult: selon le cas, ce champ contient le résultat de l'action réussie venant d'avoir lieu.

ActionError: ce champ donne une information en ce qui concerne une erreur ou une difficulté si l'action n'a pas pu être menée à bien.

Timer: ce paramètre contient la valeur particulière de la période de temporisation en attendant une réponse. Il est mis à T_1 pour le MRVT, à T_2 pour le SRVT ou à T_c pour le CVT.

La valeur est donnée dans la Recommandation Q.753.

6.2.3.2 Procédures pour action confirmée

6.2.3.2.1 Réception de demande OM-CONFIRMED-ACTION

La procédure confirmedAction est lancée à la réception à la primitive de demande OM-CONFIRMED-ACTION. Dans ce cas, l'OMPM construit une APDU demandant l'opération confirmedAction et transmet l'APDU au moyen des services TC-INVOKE et TC-BEGIN.

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

- Opération – Prend la valeur de confirmedAction.
- Classe – La valeur est 1.
- Paramètres – Correspondent aux paramètres de confirmedAction tels que définis après le mot clé «PARAMETER» de la définition de l'opération. La valeur «testRoute» est obtenue par dérivation à partir de CNF-ACTION de localForm de ActionTypedId depuis ActionType de ActionInfo.
- Temporisation – Est copiée du paramètre «Timer» de la demande OM-CONFIRMED-ACTION.
- Invoke ID et Dialogue ID sont copiés de la demande OM-CONFIRMED-ACTION.

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

- Dialogue ID – Comme dans TC-INVOKE.
- Adresse de destination – La CalledPartyAddress de la demande OM-CONFIRMED-ACTION.
- Adresse de départ – la CallingPartyAddress de la demande OM-CONFIRMED-ACTION.

La primitive de demande N-UNITDATA finalement émise vers le SCCP suite à la réception de ces primitives de demande TC devrait contenir le paramètre de commande de séquence avec la valeur correspondant à «séquence non garantie» et le paramètre d'option de retour ayant la valeur correspondant à «renvoyer le message en cas d'erreur». Voir 2.2.2/Q.711.

6.2.3.2.2 Réception de TC-BEGIN avec indication TV-INVOKE

Dans ce cas, et si l'APDU est correctement formée et demande l'opération confirmedAction, l'OMPM émet une primitive d'indication OM-CONFIRMED-ACTION au OMASE-User.

Si l'APDU n'est pas correctement formée, l'OMPM ignore les indications du TC.

Il convient qu'un mécanisme apparenté à celui du 3.3.4/Q.774 soit envisagé au niveau de la mise en oeuvre afin de tenir compte de toutes les difficultés sur le plan local.

Les paramètres supplémentaires que pourrait contenir l'APDU sont ignorés par l'OMPM.

6.2.3.2.3 Réception de la réponse OM-CONFIRMED-ACTION

La primitive de réponse OM-CONFIRMED-ACTION peut contenir soit le paramètre ActionResult, soit le paramètre ActionError.

Le paramètre ActionResult indique que l'exécution de l'opération a abouti et l'OMPM émet une primitive de demande TC-RESULT-L. Si l'essai était un CVT, les paramètres suivants sont inclus dans la primitive TC-RESULT-L:

- Opération – A la valeur de confirmedAction.
- Paramètres – Correspondent aux paramètres d'aboutissement de ACTIONRESULT pour le CVT.

La présence d'un paramètre ActionError indique que l'opération n'a pas abouti et l'OMPM émet une primitive de demande TC-U-ERROR avec les paramètres suivants:

- Error – Prend la valeur d'erreur appropriée dans la liste définie après le mot «ERRORS» de la définition de l'opération.
- Paramètres – Correspondent aux paramètres définis après le mot «PARAMETER» de la définition de l'erreur.

Le résultat de l'opération est transmis par l'OMPM qui émet une demande TC-END avec les paramètres Dialogue ID et Termination, ce dernier indiquant «terminaison de base».

La primitive de demande N-UNITDATA émise ensuite vers le SCCP suite à la réception de ces primitives de demande TC devrait contenir un paramètre Sequence control ayant la valeur correspondant à «séquence garantie», alors que le paramètre Return option devrait indiquer «refuser le message en cas d'erreur». Voir 2.2.2/Q.711.

6.2.3.2.4 Réception de TC-END avec indication TC-RESULT-L

Dans ce cas l'OMPM, à condition que l'APDU soit correctement formée, émet une primitive de confirmation OM-CONFIRMED-ACTION avec le paramètre ActionResult a la valeur de OMASE-User (Dialogue ID inclus). Si l'APDU n'est pas correctement formée, l'OMPM ignore les primitives TC.[3)]

6.2.3.2.5 Réception de TC-END avec indication TC-U-ERROR

Si l'APDU est correctement formée, l'OMPM émet une primitive de confirmation OM-CONFIRMED-ACTION avec le paramètre ActionError (et Dialogue ID) vers OMASE-USER.

Si l'APDU n'est pas correctement formée, l'OMPM ignore les primitives TC.[3)]

6.2.3.2.6 Réception de l'indication TC-L-CANCEL

Cela survient à l'expiration de la temporisation d'appel.

Lorsque l'OMPM émet une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique «failure» pour CNF-ACTION, et si l'opération invoquée est testRoute, le paramètre failureType indique timerExpired.

L'OMPM termine le dialogue avec une primitive de demande TC-END et indique le paramètre Termination indiquant «terminaison prédéterminée».

6.2.3.2.7 Réception de TC-BEGIN ou TC-END avec indication TC-L-REJECT

La réception de TC-BEGIN avec indication TC-L-REJECT est illustrée à la Figure 2a.

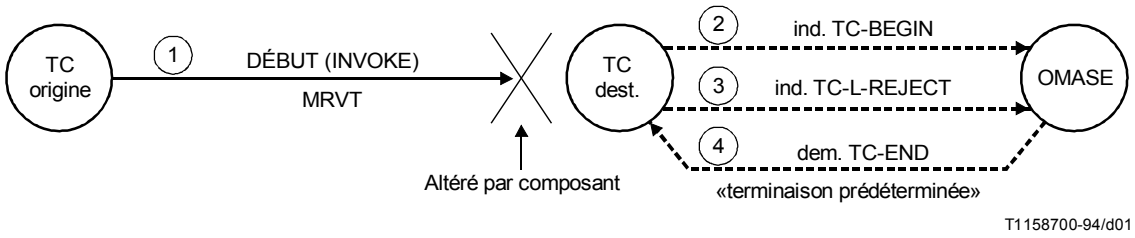

FIGURE 2a/Q.754

Si l'OMPM reçoit une indication TC-L-REJECT avec l'indication TC-BEGIN, l'OMPM met fin au dialogue en émettant une primitive de demande TC-END avec le paramètre Termination correspondant à «terminaison prédéterminée».

Si l'OMPM reçoit une indication TC-L-REJECT avec l'indication TC-END, il émet une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique «failure» de CNF-ACTION; si testRoute a été invoqué, le paramètre failureType de la primitive de confirmation indique «routeInaccessible».

Cela est illustré par la Figure 2b.

6.2.3.2.8 Réception de TC-END avec indication TC-R-REJECT

Dans ce cas, l'OMPM émet une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique «failure» de CNF-ACTION; si testRoute a été invoqué, le paramètre failureType de la primitive de confirmation indique «routeInaccessible».

6.2.3.2.9 Réception de l'indication TC-P-ABORT

Cela est illustré par la Figure 2c.

---

3) L'utilisation de la temporisation de protection globale dans OMASE-User au nœud de l'initiateur de l'essai permet, dans ces circonstances, à l'essai d'échouer de manière progressive.

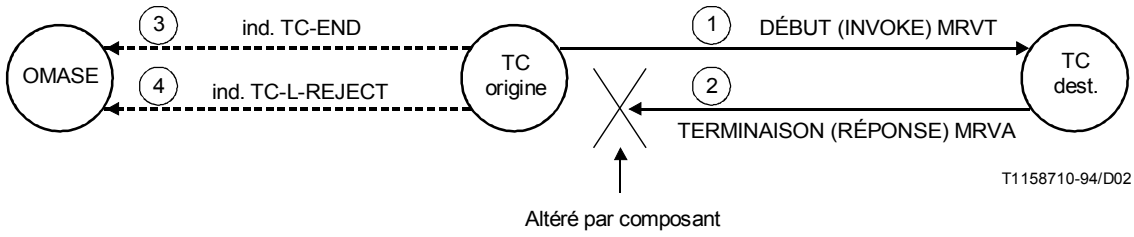

FIGURE 2b/Q.754

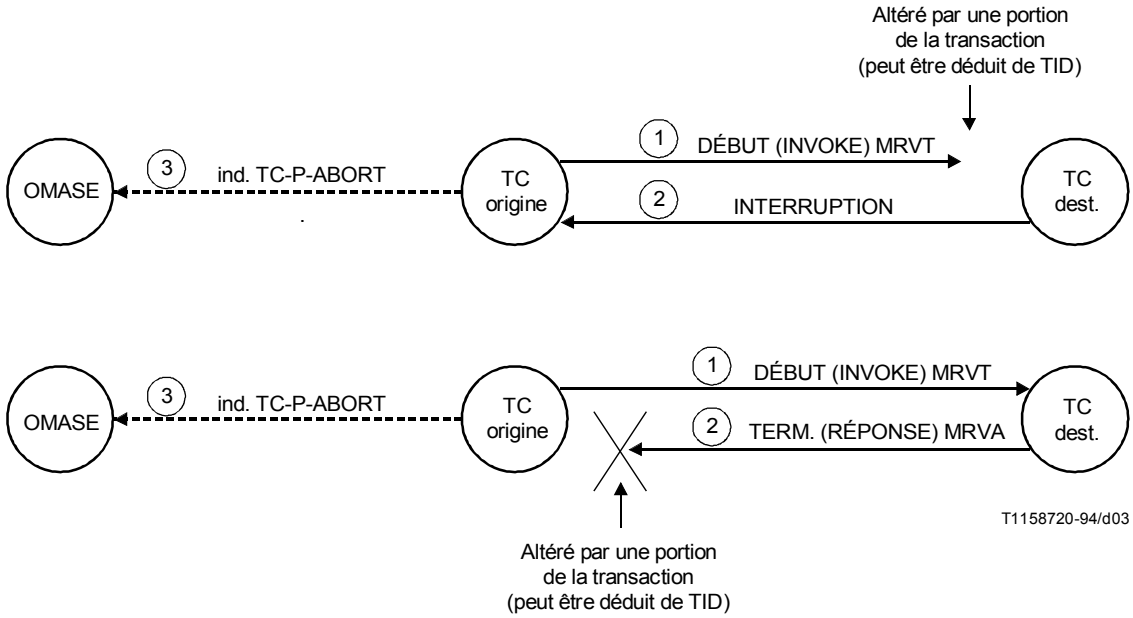

FIGURE 2c/Q.754

Dans ce cas, l'OMPM émet une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique «failure» de CNF-ACTION; si testRoute a été invoqué, le paramètre failureType de la primitive de confirmation indique «routeInaccessible».

6.2.3.2.10 Réception de TC-NOTICE

Dans ce cas, l'OMPM émet une primitive de confirmation OM-CONFIRMED-ACTION avec l'erreur spécifique «failure» de CNF-ACTION; si testRoute a été invoqué, le paramètre failureType de la primitive de confirmation indique «routeInaccessible».

Définition des erreurs

Plusieurs types d'erreur ont été utilisés dans la définition des deux services OM. Ces erreurs sont définies dans ce paragraphe.

Définitions

noSuchObjectClass: la classe d'objet de l'APDU d'appel n'est pas reconnue par l'extrémité de réception.

noSuchObjectInstance: bien que la classe d'objet de l'APDU d'appel soit connue, il n'y a pas d'instance d'objet correspondante de cette classe de primitive à l'extrémité de réception.

accessDenied: l'accès à la ressource n'est pas autorisé.

processingFailure: une défaillance a eu lieu pendant le traitement d'une action spécifique ou d'un événement. Les indicateurs de défaillance sont spécifiques à l'action ou à l'événement.

noSuchAction: ce type d'action n'est pas offert par l'extrémité de réception ou connu d'elle.

noSuchArgument: l'argument spécifié n'est pas connu de l'extrémité de réception ou n'est pas offert par elle.

invalidArgumentValue: la valeur de l'argument n'est pas appropriée pour l'extrémité de réception.

6.3 Syntaxe abstraite du protocole OMASE

```
-- protocole OMASE
OMASE { ccitt recommendation q 754 omase(0) versionl(1) }

DEFINITIONS ::=
BEGIN
-- définitions des TCAP
EXPORTS EVENT, CNF-ACTION, SPECIFIC-ERROR;

IMPORTS OPERATION, ERROR FROM TCAPMessage { ccitt recommendation q 773 modules(0) messages(1)
version2(2) };
-- opérations OMASE

        eventReport OPERATION

        PARAMETER SEQUENCE {

            managedObjectClass      ObjectClass,
            managedObjectInstance   ObjectInstance,
            eventTime               [5] IMPLICIT GeneralizedTime OPTIONAL,
            eventType               [7] IMPLICIT EVENT,
            eventInfo               [8] ANY DEFINED BY eventType OPTIONAL }

        ::= localValue 0

        confirmedAction OPERATION

        PARAMETER SEQUENCE {

                COMPONENTS OF       BaseManagedObjectId,
                accessControl       [5] AccessControl OPTIONAL,
                actionInfo          [12] IMPLICIT ActionInfo }

        RESULT actionResult         ActionResult

        ERRORS { accessDenied, invalidArgumentValue,
                noSuchAction, noSuchArgument,
                noSuchObjectClass, noSuchObjectInstance,
                processingFailure }

        ::= localValue 7
```

FIGURE 3/Q.754 (feuillet 1 sur 5)

Syntaxe formelle des services OMASE

-- Les définitions des erreurs de service OM sont:

noSuchObjectClass ERROR
PARAMETER ObjectClass
::= localValue 0

noSuchObjectInstance ERROR
PARAMETER ObjectInstance
::= localValue 1

accessDenied ERROR
::= localValue 2

noSuchAction ERROR
PARAMETER NoSuchAction
::= localValue 9

processingFailure ERROR
PARAMETER ProcessingFailure -- *optionnel*
::= localValue 10

noSuchArgument ERROR
PARAMETER NoSuchArgument
::= localValue 14

invalidArgumentValue ERROR
PARAMETER InvalidArgumentValue
::= localValue 15

FIGURE 3/Q.754 (feuillet 2 sur 5)

Syntaxe formelle des services OMASE

-- Définitions de type support:

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

ActionInfo              ::= SEQUENCE  { actionType ActionTypeId,
                                        actionInfoArg [4] ANY DEFINED BY actionType
                                        OPTIONAL }

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

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

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

EventReportArgument     ::= SEQUENCE  { managedObjectClass ObjectClass,
                                        managedObjectInstance ObjectInstance,
                                        eventTime [5] IMPLICIT GeneralizedTime
                                        OPTIONAL,
                                        eventType EventTypeId,
                                        eventInfo [8] ANY DEFINED BY eventType
                                        OPTIONAL }

EventTypeId             ::= CHOICE    {
                                        -- globalForm ...
                                        localForm [7] IMPLICIT EVENT }
```

FIGURE 3/Q.754 (feuillet 3 sur 5)

Syntaxe formelle des services OMASE

```
InvalidArgumentValue    ::= CHOICE      { actionValue                   [0] IMPLICIT ActionInfo,
                                          eventValue                    [1] IMPLICIT SEQUENCE {
                                          eventType                     EventTypeId,
                                          eventInfo                     [8] ANY DEFINED BY 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     -- défini par classe d'objet

SpecificErrorinfo       ::= SEQUENCE    { errorType                     [0] IMPLICIT SpecificError,
                                          errorParm                     [1] ANY DEFINED BY errorType
                                                                        OPTIONAL }

Timer                   ::= INTEGER     -- secondes
```

FIGURE 3/Q.754 (feuillet 4 sur 5)

Syntaxe formelle des services OMASE

```
-- Les rapports d'événements spécifiques sont classés par classe d'objet. Les utilisations du protocole
-- peuvent être décrites par le MACRO EVENT ci-dessous.

        EVENT MACRO ::=

        BEGIN

            TYPE NOTATION       ::=   EventInfo
            VALUE NOTATION      ::=   value(VALUE INTEGER)
            EventInfo           ::=   "EVENTINFO" NamedType | empty
            NamedType           ::=   identifier type | type

        END
-- Les actions spécifiques sont classées par classe d'objet. Les utilisations du protocole peuvent être décrites
-- par le macro CNF-ACTION ci-dessous.

        CNF-ACTION MACRO ::=

        BEGIN

            TYPE NOTATION       ::=   ActionArg ActionResult SpecificErrors
            VALUE NOTATION      ::=   value(VALUE INTEGER)
            ActionArg           ::=   "ACTIONARG" NamedType | empty
            ActionResult        ::=   "ACTIONRESULT" NamedType | empty
            SpecificErrors      ::=   "SPECIFICERRORS" "{" SpecificErrorList"}" | empty
            NamedType           ::=   identifier type | type
            SpecificErrorList   ::=   SpecificError | SpecificErrorList","SpecificError
            SpecificError       ::=   value(SPECIFIC-ERROR)

        END
-- Les erreurs qui sont spécifiques à une action ou à un événement sont définies au moyen du macro SPECIFIC-ERROR
-- ci-dessous.

            SPECIFIC-ERROR MACRO ::=

            BEGIN

                TYPE NOTATION       ::=   ProcessingErrorParm
                VALUE NOTATION      ::=   value(VALUE INTEGER)
                ProcessingErrorParm ::=   "PARAMETER" NamedType | empty
                NamedType           ::=   identifier type | type

            END

END -- protocole OMASE
```

FIGURE 3/Q.754 (feuillet 5 sur 5)

Syntaxe formelle des services OMASE

Annexe A

(Cette annexe fait partie intégrante de la présente Recommandation)

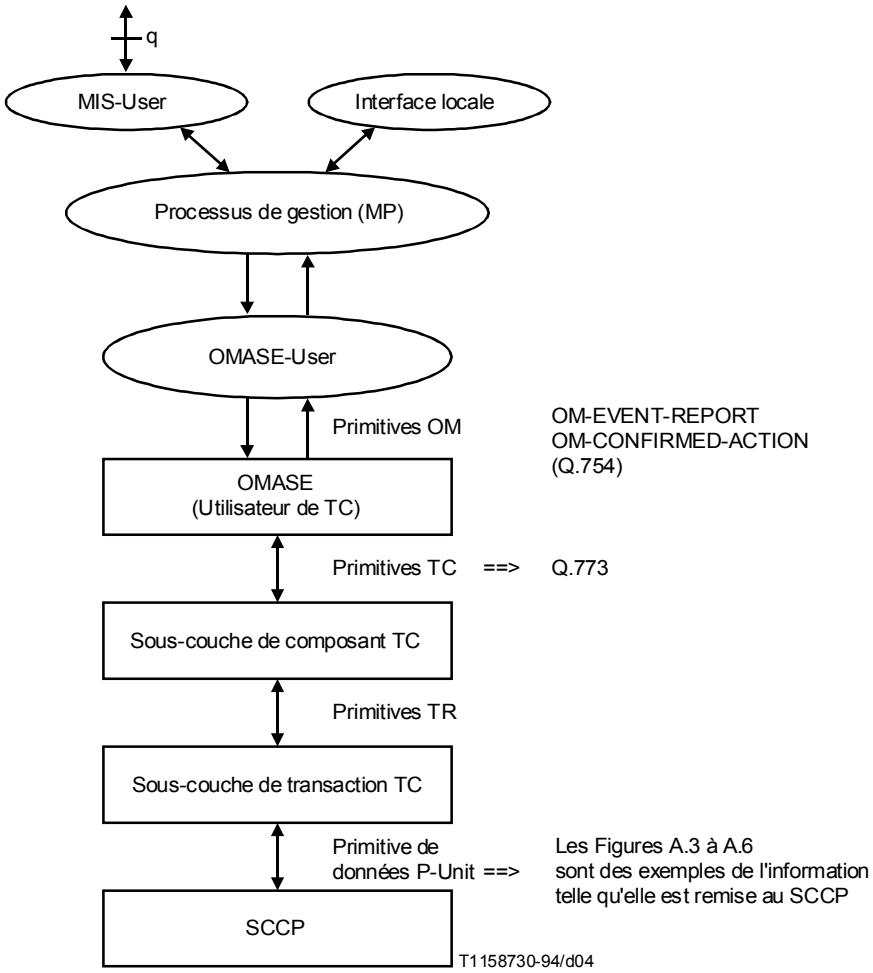

FIGURE A.1/Q.754

Interface de primitives

La Figure A.2 illustre l'utilisation des primitives dans un MRVT. Quand l'OMASE-User d'origine reçoit une demande «sendMRVT» du processus de gestion (MP – voir la Recommandation Q.753 en ce qui concerne le modèle utilisé), il construit une demande OM-CONFIRMED-ACTION. La séquence est ensuite telle qu'elle est représentée par la primitive et la séquence de messages jusqu'au numéro 5. A ce point, et si le nœud n'est pas la destination soumise à l'essai, l'OMASE-User recevant l'indication OM-CONFIRMED-ACTION demande à OM-CONFIRMED-ACTION de OMASE d'envoyer des messages MRVT sur toutes les voies jusqu'à la destination d'essai dans la table d'acheminement. Lorsque tous les messages MRVA sont reçus (considérés dans l'OMASE-User comme des primitives de confirmation OM-CONFIRMED-ACTION), l'OMASE-User émet la primitive de réponse OM-CONFIRMED-ACTION comme indiqué à l'article 6.

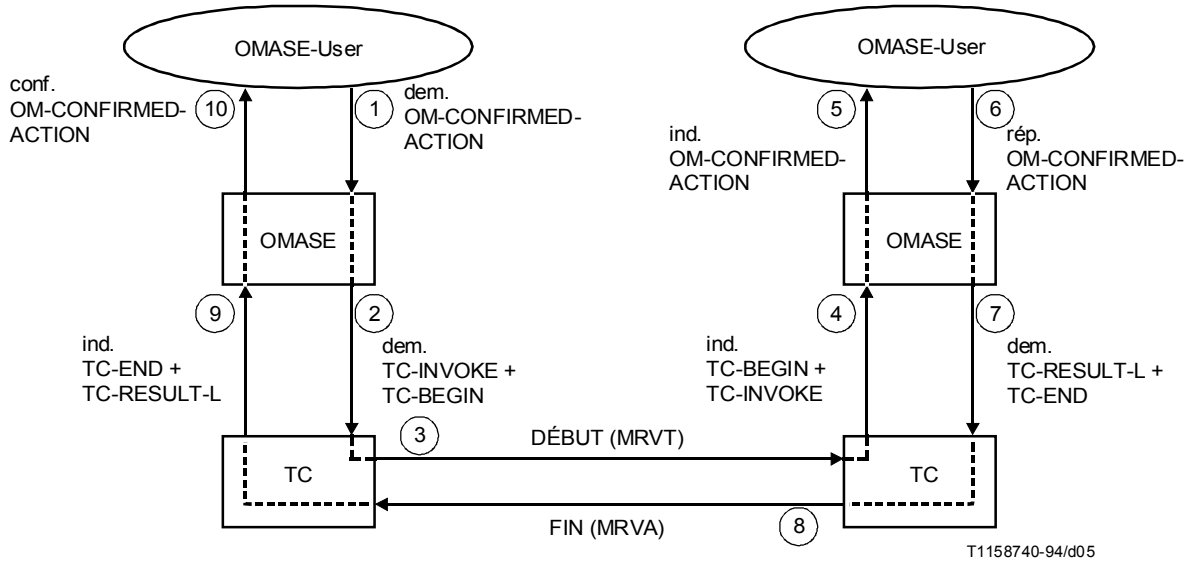

FIGURE A.2/Q.754

Exemple de l'utilisation des interfaces de primitives

Désignation du champ	Codage binaire	Référence/explication
Etiquette de type de message	01100010	= Etablissement (Tableau 8/Q.773)
Longueur du message	00110010	50 octets suivant la partie TC
Etiquette de l'identificateur de transaction	01001000	= Origine (Tableau 10/Q.773)
Longueur	00000100	4 octets
Valeur de l'identificateur de transaction	xxxxxxxx	TCAP fondé sur un dialogue au niveau
	xxxxxxxx	usager
	xxxxxxxx	
	xxxxxxxx	
Etiquette de la partie composant	01101100	(Tableau 14/Q.773)
Longueur	00101010	Les 42 octets qui suivent
Etiquette de type de composant	10100001	= Invocation (Tableau 19/Q.773)
Longueur	00101000	Les 40 octets qui suivent
Etiquette de l'identificateur de composant	00000010	= Identificateur de lancement (Tableau 20/Q.773)
Longueur	00000001	1 octet
Valeur de l'identificateur d'invocation	xxxxxxxx	FOURNI PAR OMAP
Etiquette de code d'opération	00000010	= Local. (Tableau 22/Q.773)
Longueur	00000001	1 octet
Code d'opération	00000111	= Action confirmée (Figure 3/Q.754)
Etiquette de séquence de paramètres	00110000	= Etiquette de séquence (Tableau 23/Q.773)
Longueur	00100000	Les 32 octets qui suivent
Etiquette de classe d'objet	10000000	globalForm X.711 et X.209
Longueur	00000101	5 octets
Valeur – Tables d'acheminement MTP	00000000	CCITT, Rec.
	00010001	q
	10000101	85 = > 754
	01110010	72 = >
	00000000	Tables d'acheminement MTP 1992
Etiquette d'instance d'objet	10000011	nonSpecificForm X.711 et X.209
Longueur	00000010	2 octets
Valeur d'instance d'objet	xxxxxxxx	(OMAP) destination d'essai
	xxxxxxxx	
Etiquette d'information d'action	10101100	X.711 et X.209
Longueur	00010011	Les 19 octets qui suivent
Etiquette de type d'action	10000011	localForm X.711 et X.209
Longueur	00000001	1 octet
CNF-ACTION	00000001	= voie d'essai (8.1.1/Q.754)
Etiquette Arg info Action	10100100	X.711 et X.209
Longueur	00001110	Les 14 octets qui suivent
Etiquette de séq. de paramètres	00110000	= Etiquette de séquence (Tableau 23/Q.773)
Longueur	00001100	Les 12 octets qui suivent
Etiquette de SP initiateur	10000000	8.1.1.1.1/Q.754, X.209
Longueur	00000010	2 octets
Valeur SP initiateur	xxxxxxxx	Initiateur d'essai (OMAP)
	xxxxxxxx	
Etiquette dem. trace	10000001	8.1.1.1.2/Q.754, X.209
Longueur	00000001	1 octet
Valeur	00000001	= VRAI
Etiquette de seuil	10000010	= seuil (8.1.1.1.3/Q.754)
Longueur	00000001	1 octet
Valeur seuil	xxxxxxxx	FOURNI PAR OMAP
Etiquette des codes de points traversés	10100011	8.1.1.1.4/Q.754
Longueur	00000000	Liste des codes de point vide

FIGURE A.3/Q.754

Exemple d'un message MRVT remis au SCCP

Désignation du champ	Codage binaire	Référence/Explication
Etiquette de type de message	01100100	= TERMINAISON (Tableau 8/Q.773)
Longueur de message	00001101	13 octets qui suivent dans la partie TC
Etiquette d'identificateur de transaction	01001001	= Destination (Tableau 10/Q.773)
Longueur	00000100	4 octets
Valeur d'identificateur de transaction	xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx	Le même que dans ÉTABLISSEMENT (message MRVT)
Etiquette de partie composant	01101100	(Tableau 14/Q.773)
Longueur	00000101	Les 5 octets qui suivent
Etiquette de type composant	10100010	= Résultat complet (Tableau 19/Q.773)
Longueur	00000011	Les 3 octets qui suivent
Etiquette d'identificateur de composant	00000010	= Ident. d'invocation (Tableau 20/Q.773)
Longueur	00000001	1 octet
Valeur d'identificateur d'invocation	xxxxxxxx	Comme pour le message MRVT (Corrélation)

FIGURE A.4/Q.754

Exemple d'un message MRVA (succès) remis au SCCP

Désignation du champ	Codage binaire	Référence/Explication
Etiquette de type de message	01100100	= TERMINAISON (Tableau 8/Q.773)
Longueur de message	00100000	32 octets qui suivent dans la partie TC
Etiquette d'identificateur de transaction	01001001	= Destination (Tableau 10/Q.773)
Longueur	00000100	4 octets
Valeur d'identificateur de transaction	xxxxxxxx	comme dans ÉTABLISSEMENT (message MRVT)
	xxxxxxxx	
	xxxxxxxx	
	xxxxxxxx	
Etiquette de partie composant	01101100	(Tableau 14/Q.773)
Longueur	00011000	Les 24 octets qui suivent
Etiquette de type de composant	10100011	= Résultat complet (Tableau 19/Q.773)
Longueur	00010110	Les 22 octets qui suivent
Etiquette d'identificateur de composant	00000010	= Ident. d'invocation (Tableau 20/Q.773)
Longueur	00000001	1 octet
Valeur d'identificateur d'invocation	xxxxxxxx	comme pour message MRVT (corrélation)
Etiquette de code d'erreur	00000010	Tableau 24/Q.773 (local)
Longueur	00000001	1 octet
Echec de traitement	00001010	Figure 3/Q.754
Etiquette de séquence de paramètres	00110000	= Etiquette de séquence (Tableau 23/Q.773)
Longueur	00001110	Les 14 octets qui suivent
Etiquette d'info. d'erreur spécifique	10100101	Figure 3/Q.754
Longueur	00001100	Les 12 octets qui suivent
Etiquette de type d'erreur	10000000	Figure 3/Q.754
Longueur	00000001	1 octet
Echec	00000001	2.1.1.3.1/Q.754
Paramètres d'erreur	10100001	2.1.1.3.1/Q.754
Longueur	00000111	Les 7 octets qui suivent
Etiquette de type d'échec	10000000	2.1.1.3.1/Q.754
Longueur	00000010	2 octets
Bits inutilisés	00000000	aucun
Chaîne d'échec	xxxxxxxx	Dépend du type déchec (2.1.1.3.1/Q.754)
Etiquette de trace envoyée	10000001	2.1.1.3.1/Q.754
Longueur	00000001	1 octet
Valeur de trace envoyée	0000000X	Vrai = 1, Faux = 0 (2.1.1.3.1/Q.754)

FIGURE A.5/Q.754

Exemple de message MRVA (échec) remis au SCCP

Désignation du champ	Codage binaire	Référence/Explication
Etiquette de type de message	01100010	= ÉTABLISSEMENT (Tableau 8/Q.773)
Longueur de message	00101100	44 octets qui suivent dans la partie TC
Etiquette d'identificateur de transaction	01001000	= Origine (Tableau 10/Q.773)
Longueur	00000100	4 octets
Valeur de l'identificateur de transaction	xxxxxxxx	TCAP fondé sur un dialogue au niveau utilisateur
	xxxxxxxx	
	xxxxxxxx	
	xxxxxxxx	
Etiquette de partie composant	01101100	(Tableau 14/Q.773)
Longueur	00100100	Les 36 octets qui suivent
Etiquette de type de composant	10100001	= Invocation (Tableau 19/Q.773)
Longueur	00100010	Les 34 octets qui suivent
Etiquette d'identificateur de composant	00000010	= Ident. d'invocation (Tableau 20/Q.773)
Longueur	00000001	1 octet
Valeur d'identificateur d'invocation	xxxxxxxx	FOURNI PAR OMAP
Etiquette de code d'opération	00000010	= Local (Tableau 22/Q.773)
Longueur	00000001	1 octet
Code d'opération	00000000	= Rapport d'événement (Figure 3/Q.754)
Etiquette de séquence de paramètre	00110000	= Etiquette de séquence (Tableau 23/Q.773)
Longueur	00011010	Les 26 octets qui suivent
Etiquette de classe d'objet	10000000	(Figure 3/Q.754)
Longueur	00000101	5 octets
Valeurs-Tables d'acheminement	00000000	CCITT, Rec.
MTP	00010001	q
	10000101	85 = > 754
	01110010	72 = >
	00000000	Tables d'acheminement MTP 1992
Etiquette d'instance d'objet	10000011	(Figure 3/Q.754)
Longueur	00000010	2 octets
Valeur d'instance d'objet	xxxxxxxx	PC de terminaison (OMAP)
	xxxxxxxx	<Destination d'essai>
Etiquette de type d'événement	10000111	Figure 3/Q.754
Longueur	00000001	1 octet
Type d'événement	00000010	= routeTrace (2.1.2/Q.754)
Etiquette de type d'information d'événement	10101000	Figure 3/Q.754
Longueur	00001010	Les 10 octets qui suivent
Identificateur de succès	10100000	2.1.2.1.1/Q.754
Longueur	00001000	Les 8 octets qui suivent
Etiquette de code point	00000100	= OCTET STRING
Longueur	00000010	2 octets
Code point	xxxxxxxx	
	xxxxxxxx	
Etiquette de code point	00000100	= OCTET STRING
Longueur	00000010	2 octets
Code point	xxxxxxxx	
	xxxxxxxx	

FIGURE A.6/Q.754

Exemple de message MRVR (succès) remis au SCCP

Imprimé en Suisse

Genève, 1994