UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Q.775

(06/97)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE Q: COMMUTATION ET SIGNALISATION

Spécifications du système de signalisation n° 7 – Sous-système application de Gestion des Transactions

Guide d'utilisation du gestionnaire de transactions

Recommandation UIT-T Q.775

(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.775

GUIDE D'UTILISATION DU GESTIONNAIRE DE TRANSACTIONS

Résumé

La Recommandation UIT-T Q.775 révisée donne de nouvelles directives aux utilisateurs du gestionnaire de transactions (TC) en ce qui concerne les éléments de service d'application (ASE) utilisateurs du TC.

Source

La Recommandation UIT-T Q.775, 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

		Page
1	Introduction	1
1.1	Généralités	1
1.2	Environnement	2
2	Opérations	2
2.1	Définition	2
2.2	Exemples	3
2.2.1	Traitement des opérations simples	3
2.2.2	Traitement d'opérations plus compliquées	4
2.3	Fonctionnalités relatives aux composants offertes aux utilisateurs du TC	6
2.3.1	Lancement d'opération	6
2.3.2	Annulation (par l'utilisateur du TC)	7
2.3.3	Rejet (par l'utilisateur du TC)	8
2.3.4	Annulation à distance (par l'utilisateur du TC)	9
2.3.5	Réinitialisation par l'utilisateur du TC de la temporisation pour l'exécution de l'opération	11
2.4	Situations anormales relatives aux composants	13
2.4.1	Perte de composants	13
2.4.2	Duplication de composants	14
2.4.3	Arrivée hors séquence de composants	16
2.4.4	Rejet d'un composant par le TC	17
2.4.5	Expiration de la temporisation d'opération	18
3	Dialogues	18
3.1	Groupement de composants dans un message	18
3.2	Services de gestion de dialogue	20
3.2.1	Dialogue structuré	20
3.2.2	Dialogue non structuré	30
3.3	Fonctionnalités améliorées de gestion du dialogue	30
3.3.1	Aperçu général	30
3.3.2	Utilisation du contexte d'application	30
3.3.3	Transfert des données d'utilisateur	31
3.3.4	Questions de compatibilité amont	32
4	Directives pour l'élaboration des protocoles utilisateurs de TC	33
4.1	Introduction	33
4.2	Découpe fonctionnelle dans une application	33
4.2.1	Processus d'application et entité d'application	33
4.2.2	Elément de service d'application	33

Page

4.2.3 Communications entre AE/ASE homologues 34
4.3 Comment spécifier un contexte d'application 35
4.4 Comment spécifier un ASE 35
4.5 Comment spécifier les opérations et les erreurs 36
4.5.1 Généralités 36
4.5.2 Utilisation de la notation OPERATION MACRO 37
4.5.3 Utilisation de la notation ERROR MACRO 38
4.5.4 Utilisation de la notation CLASS (classe d'objets informationnels) 39
4.5.5 Classe (d'objets informationnels) ERROR CLASS 42
4.5.6 Exemples de descriptions d'opérations et d'erreurs 43
4.5.7 Passage de la notation MACRO à la notation CLASS (classe d'objets informationnels) 45
4.5.8 Attribution et gestion des codes d'opération et d'erreur 46
4.6 Spécifications de types de données 50
4.6.1 Généralités 50
4.6.2 Utilisation d'étiquettes 50
4.6.3 Instances et types 51
4.6.4 Exportation et importation de types de données 51
4.7 Comment spécifier les syntaxes abstraites 51
4.8 Règles de codage 53
5 Mappage des concepts génériques du service ROS et des services du TC 53
5.1 Aperçu général 53
5.1.1 Notation et concept applicables au modèle générique ROS 53
5.1.2 Modèle de communication 54
5.2 Réalisation du service d'opérations distantes 56
5.2.1 Services de base (pivot) 56
5.2.2 Opérations de rattachement Bind et de détachement Unbind 57
5.3 Transfert d'information 59
5.3.1 Réalisations d'association 59
5.3.2 Réalisation de transfert 59
5.4 Contexte d'application utilisant le TC 59
5.5 Syntaxes abstraites 61
5.5.1 Commande de dialogue 61
5.5.2 Syntaxes définies par l'utilisateur 61
5.6 Extension de notation 62

Recommandation Q.775

GUIDE D'UTILISATION DU GESTIONNAIRE DE TRANSACTIONS

(révisée en 1997)

1 Introduction

1.1 Généralités

La présente Recommandation a pour but de servir de guide aux utilisateurs potentiels du gestionnaire de transactions (utilisateurs du TC). Les exemples donnés sont uniquement illustratifs; ils indiquent comment une application peut utiliser le sous-système application pour la gestion des transactions (TCAP, *transaction capability application part*) et non comment le TC doit être utilisé dans tous les cas. Les bases techniques de la présente Recommandation sont les Recommandations Q.771 à Q.774; elles seront considérées comme référence en cas de désaccord avec la présente Recommandation.

Le principal objectif du TC est de fournir un support aux applications interactives dans un environnement distribué. Le TC est basé sur la notion des opérations distantes définie dans la Recommandation X.880 (ROS, *remote operations*) ainsi que sur certaines améliorations et additions spécifiques à l'environnement du système de signalisation n° 7 de manière à fournir les services attendus par les utilisateurs du TC. Les interactions entre entités d'application distribuées sont modélisées par des opérations. Une opération est lancée par une entité (origine), l'autre entité (destination) tente d'exécuter l'opération et éventuellement renvoie le résultat de cette tentative.

Le TC n'a pas pour objet de définir la sémantique d'une opération (représentée par ses nom et paramètres). Les services offerts par le TC sont indépendants d'une opération particulière. Un utilisateur du TC qui définit une application doit:

1) sélectionner les opérations (ce qui comprend la définition de la sémantique et de la syntaxe des données échangées pendant les lancements des opérations et leurs réponses);
2) sélectionner les services du TC nécessaires à la mise en œuvre de ces opérations. Ces services comprennent la gestion d'opérations individuelles ou d'opérations apparentées, attachées à une association entre utilisateurs du TC appelée dialogue;
3) définir le scénario d'application (par exemple lequel des deux homologues lance les opérations, l'ordre de l'échange des messages qui constitue le dialogue entre les utilisateurs du TC homologues et leurs réactions face à des situations anormales).

La présente Recommandation décrit le processus de sélection des opérations. Les opérations présentées dans la présente Recommandation sont fictives, elles sont uniquement utilisées à des fins d'illustration. La présente Recommandation décrit également les services offerts par le TC pour la gestion d'une opération, ou d'une suite d'opérations dans un dialogue. Par contre, la définition de suites d'opérations particulières fait partie intégrante de la définition du protocole d'application et n'entre pas dans le cadre de la présente Recommandation. Cependant, les paragraphes 4 et 5 donnent de brèves indications sur les informations qu'une spécification d'application devrait contenir.

Les services du TC sont accessibles aux utilisateurs du TC par l'intermédiaire de primitives. Ces primitives modélisent l'interface entre le TC et ses utilisateurs mais n'imposent aucune contrainte de réalisation à cette interface.

1.2 Environnement

Le TC définit le protocole de bout en bout, entre utilisateurs du TC localisés dans un réseau sémaphore n° 7. Actuellement, aucune interface standard n'est définie pour l'utilisation du TC sur un autre protocole ou réseau de base (par exemple X.25) que le SCCP du système de signalisation n° 7.

Le TC prend en considération les utilisateurs sensibles à l'aspect temps réel qui n'ont pas d'importants volumes de données à échanger. Les protocoles standards définis pour les couches 4 à 6 de l'OSI dans les Recommandations de la série X risquant, estime-t-on, de pénaliser de façon excessive ces utilisateurs, ils ne sont pas utilisés.

En conséquence, le TC ne peut mettre en œuvre tous les types d'applications; certaines d'entre elles nécessiteront toujours des services plus élaborés tels que ceux qui sont spécifiés dans les Recommandations de la série X. En outre, en vue d'aider le concepteur d'une application à choisir les moyens de la mettre en œuvre, cette Recommandation indique non seulement ce que le TC peut faire mais aussi ce qu'il ne peut pas faire.

2 Opérations

2.1 Définition

Une opération est lancée par un utilisateur du TC d'origine pour demander à un utilisateur du TC de destination d'exécuter une action donnée.

Chaque opération relève d'une classe donnée. Elle indique si la destination doit signaler seulement les succès, seulement les échecs, les succès et les échecs ou ni les succès, ni les échecs.

La classe d'une opération n'est pas signalée à l'utilisateur du TC distant au moment où l'opération est lancée; en principe, les applications aux deux extrémités savent quelle classe est utilisée pour chaque opération.

Tout comme la classe, la définition de l'opération comprend une valeur de temporisation indiquant le délai maximal dans lequel l'opération doit être terminée et le résultat signalé.

La valeur de cette temporisation est déterminé à l'échelon local; elle n'est pas transmise à l'extrémité éloignée par l'intermédiaire d'un protocole quelconque. Elle est choisie par l'utilisateur du TC lors de la définition de l'opération sur la base des prévisions du délai aller-retour d'un utilisateur du TC à un autre et des délais de traitement.

Une opération est définie par:

– son code d'opération et le type des paramètres associés à la demande d'opération;
– sa classe;
– si la classe requiert la signalisation des succès, les résultats possibles, correspondant aux exécutions réussies, sont définis par une liste de paramètres;
– si la classe requiert la signalisation des échecs, les résultats possibles, correspondant aux situations où l'opération n'a pu être complètement exécutée par l'utilisateur du TC distant, sont identifiés par des causes d'erreur spécifiques. La liste de ces causes d'erreur fait partie de la définition de l'opération. Une information de diagnostic peut être ajoutée à la cause d'erreur: si elle est présente, elle fait également partie de la définition;
– la liste des opérations corrélées éventuelles, si les réponses aux opérations corrélées sont autorisées pour cette opération. Les opérations corrélées doivent être décrites séparément;
– une valeur de temporisation indiquant le temps après lequel l'opération doit être terminée et éventuellement renvoyée. Cette valeur de temporisation peut être l'un des facteurs utilisés

par une mise en œuvre pour gérer l'identificateur de lancement associé au lancement de l'opération. (A l'expiration de la temporisation associée au lancement d'une opération, l'identificateur de lancement est restitué à l'ensemble des identificateurs de lancement après une période de "gel" appropriée qui varie selon le mode de réalisation).

En règle générale, le choix de la classe d'une opération devrait reposer sur la sémantique associée à une opération et les opérations ne devraient pas être conçues de manière à être acheminées dans un message donné. Si par exemple, celui qui lance l'opération n'a pas besoin d'un accusé de réception lui permettant de savoir si l'opération peut ou non être effectivement menée à bien, une opération de classe 4 sera sans doute indiquée. S'il n'a pas besoin de savoir explicitement qu'une opération a été réalisée avec succès mais souhaite savoir si elle n'a pas pu être accomplie, une opération de classe 2 est justifiée. Définir par exemple une opération telle que "annonce de lecture" comme une opération de classe 2 ou 4 dénote une diversité d'intentions de la part de celui qui lance l'opération même si les actions de celui qui réalise les opérations peuvent être identiques.

2.2 Exemples

2.2.1 Traitement des opérations simples

NOTE – Le lancement de l'opération doit tenir dans un message, de même que la notification de succès. Les notifications de succès peuvent être fractionnées en une indication de résultat(s) partiel(s) Return Result – Not last et une indication de résultat complet Return Result – Last.

Classe 1 (les succès et les échecs sont signalés)

Traduire un numéro de libre appel en un numéro d'abonné demandé; renvoyer le numéro demandé si la traduction peut être réalisée, autrement indiquer pourquoi elle ne peut l'être; temps attribué: 2 secondes.

Lorsque aucune réponse n'est renvoyée à la suite d'un lancement d'opération après l'expiration de la temporisation, l'utilisateur du TC en est informé (annulation de l'opération par le TC); il peut penser que le lancement ou la réponse s'est perdu et, selon les applications, prendre les mesures correctives voulues (par exemple nouveau lancement de l'opération, information de la gestion locale, etc.).

Classe 2 (seuls les échecs sont signalés)

Réaliser un test de routine et envoyer une réponse uniquement en cas d'échec du test; temps attribué: 1 minute.

Dans une opération de classe 2, lorsque aucun résultat n'a été reçu, l'utilisateur du TC est informé par l'expiration de la temporisation. Elle est interprétée comme une exécution réussie, même en cas de perte du lancement d'opération.

Cet aspect doit être pris en compte lors du choix de la classe 2.

Classe 3 (seuls les succès sont signalés)

Réaliser un essai: cela correspond à un point de vue pessimiste, où un échec est considéré comme une option par défaut, ne nécessitant aucune réponse.

L'expiration de la temporisation est indiquée à l'utilisateur du TC: elle doit être interprétée comme un échec de l'opération (mais elle est normale pour le gestionnaire de transactions lui-même qui considère que l'opération est terminée). Cet aspect doit être pris en compte lors du choix de la classe 3.

Classe 4 (ni les succès, ni les échecs ne sont signalés)

Envoyer un avis, sans attendre de réponse ou d'accusé de réception d'aucune sorte.

Dans ce cas, aucun résultat n'est reçu suite au lancement d'opération. L'utilisateur du TC s'en remet au TC et au réseau pour délivrer le lancement. La notification de l'expiration de la temporisation est un problème local.

Comparaison avec les classes d'opérations de ROSE (Recommandation X.219)

ROSE offre cinq classes d'opérations: les classes 2 à 5, appelées classes asynchrones, sont identiques aux classes 1 à 4 du TC. La classe 1 de ROSE est une classe synchrone, autorisant les échanges bidirectionnels de composants. Elle n'a pas d'équivalent dans le TC.

Cependant, un utilisateur du TC peut décider de fonctionner en mode synchrone (voir 3.2.1).

2.2.2 Traitement d'opérations plus compliquées

Opérations avec résultats segmentés

Un résultat de succès peut être divisé en plusieurs segments, chacun d'eux est indiqué à celui qui a lancé l'opération par une primitive. Ce service, utilisant la primitive TC-RESULT-NL, peut être utilisé par les utilisateurs du TC pour pallier l'absence de segmentation des couches sous-jacentes. Le dernier segment est indiqué par la primitive TC-RESULT-L.

La signalisation d'une erreur ne peut être segmentée.

Quand le concepteur du protocole peut faire en sorte que la segmentation soit assurée dans les couches sous-jacentes sur les voies de signalisation sur lesquelles les messages du TC sont transférés, l'utilisation de ce service de segmentation est déconseillée.

Le gestionnaire de transactions ne peut identifier un segment particulier dans le cas d'un résultat segmenté.

L'utilisateur du TC doit veiller à ce que chaque segment puisse être analysé (c'est-à-dire que le paramètre Parameter de chaque primitive de demande TC-RESULT-NL/L devrait contenir suffisamment d'informations pour permettre la construction d'une valeur valable du type (ou du sous-type compatible) associé au résultat de l'opération).

Exemple E1: une opération demande l'exécution d'un essai. Le résultat d'une exécution correcte est segmenté en trois parties P1, P2 et P3 à renvoyer à celui qui a lancé l'opération.

Une séquence possible de primitives pour l'exemple E1 est donnée au Tableau 1.

Tableau 1/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (essai, classe = 1)	indication TC-INVOKE (essai) demande TC-RESULT-NL (P1)
indication TC-RESULT-NL (P1)	demande TC-RESULT-NL (P2)
indication TC-RESULT-NL (P2)	demande TC-RESULT-L (P3)

Tableau 1/Q.775 (*fin*)

Utilisateur du TC A	Utilisateur du TC B
indication TC-RESULT-L (P3)	
temps	
NOTE – La valeur de temporisation est spécifiée par l'utilisateur du TC d'origine au moment du lancement. Un résultat qui n'est pas final ne la relance pas.	

Opérations corrélées

Une autre extension des procédures de base est la possibilité de corréler un lancement d'opération à un autre lancement d'opération.

Typiquement, ce service couvre les situations où la destination de l'opération initiale (ou corrélée à l'opération initiale) requiert des informations supplémentaires afin de traiter cette opération: c'est le cas lorsque des services en mode menu sont utilisés (les services en mode menu permettent à un utilisateur de faire une suite de choix, chacun étant dépendant du précédent).

Exemple E2: l'opération est l'exécution d'un essai à plusieurs options. Avant d'exécuter l'essai, ces options sont présentées, pour sélection, à l'initiateur de l'essai (utilisateur du TC A). Les deux opérations sont imbriquées: l'opération 1 est l'essai lui-même; l'opération 2 est la sélection de l'option. L'utilisateur du TC A répond d'abord à l'opération 2 pour que l'utilisateur du TC B puisse exécuter l'essai avec l'option ou les options indiquées.

Une séquence possible de primitives pour l'exemple E2 est donnée au Tableau 2.

Il n'y a pas de limite au nombre de lancements d'opération qui peuvent être corrélés à un lancement d'opération donné.

Il est à noter que lorsqu'une opération B est corrélée à une autre opération A, ces deux opérations n'ont pas à être imbriquées. La seule condition est que le lancement de B intervienne avant que le résultat de A ne soit reçu; cependant, l'opération B n'a pas à être terminée avant l'opération A.

Tableau 2/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (essai, classe = 1)	début de l'opération 1
	indication TC-INVOKE (essai)
	demande TC-INVOKE (sélection d'option classe = 1) début de l'opération 2

Tableau 2/Q.775 (*fin*)

Utilisateur du TC A	Utilisateur du TC B	
indication TC-INVOKE (sélection d'options) demande TC-RESULT-L (options)	indication TC-RESULT-L (options) demande TC-RESULT-L (résultat de l'essai)	fin de l'opération 2
indication TC-RESULT-L (résultat de l'essai)		fin de l'opération 1
temps		

2.3 Fonctionnalités relatives aux composants offertes aux utilisateurs du TC

2.3.1 Lancement d'opération

Jusqu'ici, les opérations ont été considérées d'un point de vue statique. Le lancement introduit un aspect dynamique: un lancement d'opération particulier doit être différencié des autres lancements éventuels en cours pour la même opération ou pour une autre.

Chaque activation particulière d'une opération est identifiée par un identificateur de lancement. Cet identificateur doit être sans ambiguïté. Il est choisi par l'utilisateur du TC qui initie le lancement d'opération et est transmis à l'utilisateur du TC de destination, lequel le réutilisera dans sa réponse (ou dans chaque segment de sa réponse) ou dans un lancement corrélé, établissant ainsi une corrélation entre la réponse à un lancement (ou chaque segment d'une réponse) ou un lancement corrélé et le lancement lui-même.

L'utilisateur du TC est libre d'attribuer n'importe quelle valeur à l'identificateur de lancement (index, adresse, etc.) à condition que cette valeur mappe à un nombre entier qui puisse être codé dans un octet conformément aux règles de codage spécifiées dans la Recommandation Q.773. Il convient de remarquer que la valeur de ce nombre entier est comprise entre −128 et 127.

L'identificateur de lancement associé à un lancement est à nouveau utilisable lorsqu'un résultat complet ou partiel est reçu, ou lorsque certaines situations anormales sont signalées par le TC. Cependant, la valeur ne doit pas être immédiatement réattribuée pour une autre activation d'opération, car cela pourrait nuire à la gestion correcte de certaines situations anormales (voir ci-dessous).

La période durant laquelle un identificateur de lancement est libre, mais inutilisable, est dite gelée. Cette période varie selon le mode de réalisation.

Comme les identificateurs de lancement reçoivent leur valeur dynamiquement lors du lancement de l'opération, cette valeur ne peut apparaître dans la spécification des protocoles d'application. Elle est plutôt une valeur "logique", à laquelle une valeur réelle est substituée au moment de l'exécution afin d'identifier une opération dans un flot simple.

En tenant compte des identificateurs de lancement, la séquence de primitives de l'exemple E2 devient comme indiqué au Tableau 3:

Tableau 3/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (1, essai, classe = 1)	indication TC-INVOKE (1, essai) demande TC-INVOKE (2, 1, sélection d'options, classe = 1)
indication TC-INVOKE (2, 1, sélection d'options) demande TC-RESULT-L (2, options)	indication TC-RESULT-L (2, options) demande TC-RESULT-L (1, résultat de l'essai)
indication TC-RESULT-L (1, résultat de l'essai)	
temps	

où le premier paramètre d'une primitive indique l'identificateur de lancement. Lorsque deux paramètres sont présents, le second est l'identificateur de corrélation. Ceci est une pure convention de notation.

2.3.2 Annulation (par l'utilisateur du TC)

L'utilisateur du TC demandant le lancement d'une opération peut arrêter, lorsqu'il le juge nécessaire, l'activité associée à l'identificateur de lancement correspondant. Cependant, l'annulation devrait, en principe, être réservée aux situations anormales: la méthode normale pour terminer une opération est la réception d'un résultat ou l'expiration de la temporisation.

L'annulation a un effet uniquement local: elle n'interdit pas à un utilisateur du TC distant d'envoyer des réponses à une opération annulée. Le TC rejettera ces réponses, lors de leur réception. Cela est décrit ci-dessous dans une séquence de primitives de l'exemple E1, où l'utilisateur du TC A annule l'essai après réception du premier résultat partiel.

Dans le Tableau 4, le segment P2 de la réponse n'est pas reçu par l'utilisateur du TC A: le TC détecte une situation de rejet (identificateur de lancement inactif) et ne le remet donc pas à l'utilisateur du TC A. Par ailleurs, toute tentative de l'utilisateur du TC B d'envoyer de nouveaux segments de la réponse est rejetée en A.

Tableau 4/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (1, essai, classe = 1)	indication TC-INVOKE (1, essai) demande TC-RESULT-NL (1, P1)

Tableau 4/Q.775 (*fin*)

Utilisateur du TC A	Utilisateur du TC B
indication TC-RESULT-NL (1, P1) décision d'annulation: demande TC-CANCEL (1) indication TC-REJECT (1, code de problème) ...	 demande TC-RESULT-NL (1, P2) ...
temps	

2.3.3 Rejet (par l'utilisateur du TC)

Il appartient à l'utilisateur du TC de décider du moment où il envoie un composant de rejet ou retourne une indication d'erreur (incapacité d'exécuter une opération). L'utilisateur du TC peut rejeter un composant lorsqu'il le juge nécessaire, à condition toutefois qu'un code de rejet approprié soit défini dans les spécifications du TC (par exemple, élément d'information obligatoire manquant dans une opération, erreur de réponse, opération inattendue, opération inconnue, etc.) et qu'il puisse l'utiliser.

De même, l'utilisateur du TC choisit le code d'erreur et l'information de diagnostic (qui est spécifiée dans le cadre des spécifications de protocole utilisateur du TC et acceptée par les deux utilisateurs du TC homologues uniquement à cette fin) qu'il utilise lorsqu'il envoie un composant d'erreur.

Le rôle de l'utilisateur du TC est illustré dans l'exemple ci-après.

L'utilisateur du TC en A s'attend, dans une situation donnée, à recevoir l'opération Y uniquement comme une opération corrélée. Lorsqu'il reçoit de B un composant de lancement avec un code d'opération se rapportant à Y mais pas d'identificateur corrélé, l'utilisateur du TC en A peut choisir:

– de ne pas exécuter l'opération et de renvoyer une erreur avec un paramètre de diagnostic déterminé au préalable dans les spécifications d'application de l'utilisateur du TC uniquement à cette fin;
– de rejeter le composant comme une "opération non reconnue".

L'interprétation du diagnostic d'erreur ou du code du problème de rejet renvoyé relève de l'utilisateur du TC en B et n'est pas décrite dans les Recommandations relatives au TC.

Le rejet d'un lancement d'opération, ou d'un résultat, affecte l'opération dans sa totalité: aucune réponse ultérieure ne sera acceptée par le TC pour ce lancement. Le rejet d'une opération corrélée n'affecte pas l'opération à laquelle elle est corrélée en ce qui concerne le TC. Les utilisateurs de TC devraient décrire leurs réactions face à des situations anormales de ce type dans leur programme d'application.

Cela est illustré par le Tableau 5 où, dans l'exemple E2, l'utilisateur du TC A ne prévoit pas le traitement de sélection d'options (qui est peut-être une caractéristique optionnelle), et rejette l'opération avec le code de problème "opération corrélée inattendue". L'utilisateur du TC B peut alors décider d'exécuter l'essai en supposant une option par défaut.

Tableau 5/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (1, essai, classe = 1)	indication TC-INVOKE (1, essai) demande TC-INVOKE (2, 1, sélection d'options, classe = 1)
indication TC-INVOKE (2, 1, sélection d'options) demande TC-U-REJECT (2, code de problème)	
	indication TC-U-REJECT (2, code de problème) demande TC-RESULT-L (1, résultat de l'essai)
indication TC-RESULT-L (1, résultat de l'essai)	
temps	

Lorsqu'un lancement d'opération est rejeté, l'utilisateur du TC peut décider d'un nouveau lancement avec un nouvel identificateur de lancement (par exemple, parce que le composant de lancement a été altéré). Il peut également décider d'abandonner le dialogue. Un dialogue très simple (question-réponse) peut ne pas définir de mécanismes de rétablissement, sauf si l'opération est d'une importance critique (par exemple, mise à jour de base de données).

2.3.4 Annulation à distance (par l'utilisateur du TC)

Le TC n'assure aucun service spécifique pour l'annulation à distance de l'exécution d'une opération en cours. Le service d'annulation prévu par la primitive de demande TC-U-CANCEL n'a qu'un effet local (voir 2.3.2).

Une procédure d'annulation à distance peut toutefois être définie au niveau de l'utilisateur du TC avec les services du TC existants. Une solution pour l'utilisateur du TC consiste à inclure dans l'un des éléments ASE utilisés par le contexte d'application une opération dont le but est d'annuler les lancements existants.

Le module de la notation ASN.1 ci-après décrit ce type d'opération. Ce type (et celui d'erreur corrélé) peut être importé dans l'un des modules utilisés par l'utilisateur du TC de sorte qu'une opération et des codes d'erreur adaptés puissent être attribués. L'utilisateur a aussi la possibilité de concevoir sa propre opération d'après les mêmes principes.

```
TCAP-Tools { ccitt recommendation q 775 modules(2) tools(1) version1(1) }

DEFINITIONS ::=
BEGIN
EXPORTS Cancel, CancelFailed, Cancelled;

IMPORTS OPERATION, ERROR, InvokeIdType
FROM TCAPMessages { ccitt recommendation q 773 modules(2) messages(1) version2(2) };
```

```
Cancel ::= OPERATION
ARGUMENT InvokeIdType
-- un utilisateur de TC peut redéfinir ce type pour inclure un résultat vide afin d'obtenir une opération
-- de classe 1
ERRORS { CancelFailed }
-- temporisateur = 15 s

CancelFailed ::= ERROR
PARAMETER SET {
        problem  [0] CancelProblem,
        invokeId  [1] InvokeIdType }

CancelProblem ::= ENUMERATED
{ unknownInvocation(0),
        tooLate (1),
        notCancellable (2) }
-- un utilisateur de TC peut redéfinir ce type pour inclure des problèmes spécifiques à l'application

Cancelled ::= ERROR
-- une erreur de ce type doit être incluse dans la liste d'erreurs des opérations annulables

END
```

Il faut inclure une erreur "annulé" dans la liste des erreurs relatives aux opérations qui peuvent être annulées. Dans ce cas, l'utilisateur du TC qui reçoit la demande d'annulation émet une primitive de demande TC-U-ERROR pour mettre fin à l'opération. A la réception d'un composant d'erreur comprenant ce code d'erreur et d'une primitive d'indication TC-U-ERROR correspondante, il est mis fin à l'opération du côté du lancement.

L'opération qu'il convient d'annuler est identifiée par l'identificateur de lancement qui a été attribué au moment du lancement. Le lancement de l'opération d'annulation n'affecte pas le système d'état de lancement de l'opération qui doit être annulée car l'identificateur de lancement contenu dans l'argument de l'opération n'est pas visible par la sous-couche du composant.

Si l'annulation échoue, une erreur d'utilisateur est signalée avec trois diagnostics possibles:

– lancement inconnu (unknownInvocation): le lancement n'a jamais été fait ou a été oublié;

– trop tard (tooLate): le lancement est toujours connu mais l'exécution en est à un stade qui ne permet pas une annulation;

– annulation impossible (notCancellable): l'identificateur de lancement dans l'argument de l'opération d'annulation correspond à une opération que les utilisateurs de TC jugent impossible à annuler par l'initiateur du lancement.

Lorsque l'annulation réussit et qu'aucune erreur de retour avec un code d'erreur indiquant "annulé" n'est reçue pour l'opération annulée, la temporisation associée à l'opération annulée expire, d'où l'émission par le TC d'une primitive d'indication TC-L-CANCEL.

L'utilisateur du TC qui demande l'annulation peut aussi décider d'émettre une primitive de demande TC-U-CANCEL immédiatement après avoir transmis la demande d'annulation de sorte qu'il n'existe pas d'autres activités locales pour l'opération qui doit être annulée.

L'utilisation du mécanisme d'annulation a un sens si l'opération d'annulation est lancée avant l'expiration de la temporisation associée à l'opération qu'il convient d'annuler. Après, l'identificateur de lancement risque de ne pas être reconnu du côté de l'exécution car l'opération peut être déjà exécutée et la primitive de réponse transmise. La possibilité d'annuler l'exécution d'une opération après l'expiration de la temporisation ne relève pas des présentes directives car cela signifierait que

l'objectif n'est pas d'annuler l'exécution de l'opération mais les actions ultérieures qui peuvent avoir été provoquées du côté de l'exécution par le lancement.

2.3.5 Réinitialisation par l'utilisateur du TC de la temporisation pour l'exécution de l'opération

La sélection d'une valeur de temporisation appropriée fait partie intégrante de la définition d'une opération. La surveillance de la temporisation est déclenchée par le TC au moment où le composant de lancement auquel elle se rapporte est envoyé. Cependant, l'utilisateur du TC peut demander au TC de réinitialiser cette temporisation à tout moment avant qu'elle n'expire.

Le concepteur du protocole choisit la valeur de temporisation d'après une estimation du temps nécessaire à l'exécution de l'opération et au transfert du résultat. Cependant, un allongement sensible du temps d'exécution effectif est à prévoir dans certains cas, ce qui rend difficile le choix d'une valeur appropriée.

Pour éviter que la temporisation choisie pour l'exécution de l'opération n'expire pendant que la demande est encore en cours, le concepteur du protocole peut procéder de l'une des deux manières suivantes:

a) sélectionner une valeur de temporisation supérieure à la durée prévue d'exécution de l'opération dans le cas le plus défavorable;

b) sélectionner une valeur de temporisation correspondant au cas le plus probable et faire en sorte que l'utilisateur du TC utilise le service TC-RESET-TIMER quand il constate que l'exécution de l'opération prendra plus longtemps que prévu. Il convient de définir les procédures à appliquer dans ce cas pour informer l'utilisateur du TC d'origine de cette situation (utilisation d'opérations corrélées, par exemple).

Dans l'exemple qui suit, l'utilisateur du TC A lance plusieurs fois l'opération "interrogation" pour retrouver trois éléments d'information différents dans un système de base de données réparties dans deux emplacements B1 et B2. Ne sachant pas comment les données sont réparties dans ces deux emplacements, l'utilisateur du TC A envoie toutes les demandes à l'utilisateur du TC en B1.

Dans le premier cas, l'information demandée est disponible localement en B1. Dans les deux autres cas, l'information demandée doit être extraite de B2. L'utilisateur du TC B1 informe l'utilisateur du TC A que l'exécution de la demande prendra plus de temps que prévu et il lance une opération "attente". Dans le deuxième cas, l'utilisateur du TC A accepte d'attendre plus longtemps que prévu et demande au TC de réinitialiser la temporisation pour l'exécution de l'opération. Dans le troisième cas, l'utilisateur du TC A préfère renoncer à l'opération selon une procédure qui lui est propre, semblable aux procédures décrites au 2.3.4. Voir le Tableau 5 *bis*.

Tableau 5bis/Q.775

Utilisateur du TC A	Utilisateur du TC B1	Utilisateur du TC B2
première interrogation demande TC-INVOKE (1, interrogation, temporisation = 5 s)	indication TC-INVOKE (1, interrogation) information demandée disponible localement demande TC-RESULT-L (1, résultat de l'interrogation)	

Tableau 5bis/Q.775 (*suite*)

Utilisateur du TC A	Utilisateur du TC B1	Utilisateur du TC B2
indication TC-RESULT-L (1, résultat de l'interrogation) deuxième interrogation demande TC-INVOKE (2, interrogation temporisation = 5 s)	indication TC-INVOKE (2, interrogation) information demandée non disponible localement demande TC-INVOKE (3, 2, attente) demande TC-INVOKE (1, interrogation)	
indication TC-INVOKE (3, 2, attente) demande TC-RESET-TIMER (2)		indication TC-INVOKE (1, interrogation) demande TC-RESULT-L (1, résultat de l'interrogation)
	indication TC-RESULT-L (1, résultat de l'interrogation) envoyer résultat reçu à A demande TC-RESULT-L (2, résultat de l'interrogation)	
indication TC-RESULT-L (2, résultat de l'interrogation) troisième interrogation demande TC-INVOKE (3, interrogation, temporisation = 5 s)	indication TC-INVOKE (3, interrogation) information demandée non disponible localement demande TC-INVOKE (4, 3, attente) demande TC-INVOKE (2, interrogation)	
indication TC-INVOKE (4, 3, attente) demande TC-INVOKE [5, annulation (3)]		indication TC-INVOKE (2, interrogation) demande TC-RESULT-L (2, résultat de l'interrogation)

Tableau 5bis/Q.775 (*fin*)

Utilisateur du TC A	Utilisateur du TC B1	Utilisateur du TC B2
	indication TC-INVOKE [5, annulation (3)] demande TC-U-ERROR (3, annulé) indication TC-RESULT-L (1, résultat de l'interrogation) renoncer à envoyer le résultat reçu à A	
indication TC-U-ERROR (3, annulé)		

2.4 Situations anormales relatives aux composants

2.4.1 Perte de composants

Le TC suppose un taux de perte de messages dans le réseau très faible. Si ce taux est jugé trop élevé pour une application, celle-ci doit s'orienter vers un service réseau en mode connexion. Si certaines informations de protocole requièrent une qualité de service améliorée (par exemple, informations de taxation), l'application doit introduire ses propres mécanismes afin d'obtenir une meilleure fiabilité pour ces informations.

Perte d'un lancement d'opération

Le Tableau 6 présente le cas de l'exemple E1, où aucune réponse à l'essai n'est reçue avant l'expiration de la temporisation.

Tableau 6/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (1, essai, classe = 1)	
expiration de la temporisation: indication TC-L-CANCEL (1)	
temps	

Lorsqu'une opération de classe 1 est perdue, l'utilisateur du TC est informé par l'expiration de la temporisation associée à l'opération. Lorsqu'une opération de classe 1 avec un résultat (dans un segment unique) est perdue, le TC ne peut indiquer lequel du lancement d'opération ou de la réponse a été perdu. Si l'application a besoin d'identifier lequel de ces deux composants a été perdu, elle doit le faire par un protocole d'application (par exemple, en utilisant l'horodatage ou en accusant réception du lancement d'opération avant d'y répondre).

Pour une opération de classe 2, la perte sera considérée comme un succès (que ce soit le lancement ou l'horodatage d'échec qui ait été perdu). Vu le taux de perte, cela peut être acceptable pour des opérations non critiques (par exemple, des mesures de nature statistique).

Pour une opération de classe 3, la perte sera considérée comme un échec (que ce soit le lancement ou la notification de succès qui ait été perdu).

Pour une opération de classe 4, la perte n'est pas vue du TC.

Perte d'un résultat

– la perte d'un résultat partiel n'est jamais détectée par le TC.

– la perte d'un résultat complet sera éventuellement signalée à l'utilisateur du TC à l'expiration de la temporisation, mais ne pourra pas toujours être interprétée clairement comme la perte d'une réponse. Lorsque aucun résultat partiel n'est reçu, c'est peut-être parce que le lancement d'opération a été perdu.

Perte d'une opération corrélée

La perte d'une opération corrélée a le même effet que la perte d'une opération non corrélée. Elle n'a aucun effet sur l'opération à laquelle elle est corrélée.

Perte d'un composant de rejet

Ce cas devrait être extrêmement rare; il n'y a donc pas lieu de tenter de préserver les applications de telles situations. Si la perte d'un rejet concerne un lancement d'opération, l'expiration de la temporisation informera l'utilisateur du TC qui a lancé l'opération que le lancement (ou la réponse) a été perdu, lequel réagira en conséquence. Si la perte d'un rejet concerne une réponse, l'initiateur de cette réponse ne saura pas que celle-ci est incorrecte; il appartiendra à l'initiateur de l'opération de détecter la perte.

2.4.2 Duplication de composants

La duplication des messages étant très rare dans le réseau du système de signalisation n° 7, il n'est pas nécessaire d'élaborer pour les applications des scripts complexes destinés à remédier à de telles situations. Cependant, chaque fois que la duplication est inacceptable pour une application, celle-ci devra disposer de ses propres mécanismes de détection ou utiliser un service réseau en mode connexion.

Duplication d'un lancement d'opération

Lorsqu'un lancement d'opération est dupliqué (par le fournisseur de service), l'utilisateur du TC de destination (B) peut détecter ou non la duplication:

– l'utilisateur du TC B détecte la duplication: la duplication peut être rejetée avec le code de problème "identificateur de lancement dupliqué". Dans ce cas, l'utilisateur du TC distant peut interpréter le rejet comme le rejet de lancement initial.

– l'utilisateur du TC B ne détecte pas la duplication: cela peut être le cas lorsque la relation entre A et B est de type maître-esclave et que B exécute l'opération sans connaître le contexte.

En nous plaçant dans cette seconde hypothèse dans l'exemple E1, une séquence possible pourrait être celle donnée au Tableau 7.

Tableau 7/Q.775

Utilisateur du TC A	Utilisateur du TC B	
demande TC-INVOKE (1, essai, classe = 1)	indication TC-INVOKE (1, essai)	
	indication TC-INVOKE (1, essai)	duplication de lancement non détectée
	demande TC-RESULT-NL (1, P1)	
	demande TC-RESULT-NL (1, P1)	
indication TC-RESULT-NL (1, P1)		
indication TC-RESULT-NL (1, P1)		
A détecte une situation anormale et rejette: demande TC-U-REJECT (1, code de problème)	demande TC-RESULT-NL (1, P2)	
	indication TC-U-REJECT (1, code de problème)	
Le TC détecte une situation anormale et rejette P2: indication TC-L-REJECT (1, code de problème)		
	indication TC-R-REJECT (1, code de problème)	
temps		

Dans cette séquence, l'utilisateur du TC B considère qu'il reçoit deux lancements indépendants et répond à chacun d'eux. Le premier résultat P1 est accepté, puis l'utilisateur du TC A détecte que P1 est reçu une seconde fois et le rejette, ce qui termine l'opération. Lorsque le résultat P2 est reçu, il est rejeté (par le TC), par conséquent, les deux activités à l'extrémité B prendront fin à la réception des rejets.

Duplication d'un résultat partiel

Si un résultat partiel est dupliqué (RR-L), le TC ne pourra le détecter et le délivrera deux fois à l'utilisateur du TC. La détection de cette situation est du ressort de l'application.

Duplication d'un résultat complet

Si un résultat complet est dupliqué (RR-L), le TC le détecte. Le second résultat complet est considéré comme anormal (le premier résultat complet a mis fin à l'opération), et le TC le rejette.

Le Tableau 8 montre une séquence pour l'exemple E1 où le troisième segment du résultat est dupliqué (par le réseau).

Commentaire: la destruction par le TC des composants dupliqués est certainement utile. Il faut toutefois noter que:

1) la définition de solutions plus élaborées nécessiterait un degré de complexité du TC plus important, en contradiction avec ses caractéristiques de base dans une approche sans connexion;

2) une telle situation est très rare, au moins dans le réseau sémaphore n° 7.

Pour préserver une application de ces situations, il serait préférable d'utiliser un service réseau en mode connexion, puisque les duplications seraient alors détectées et gérées par les couches inférieures.

Tableau 8/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (1, essai, classe = 1) indication TC-RESULT-NL (1, P1) indication TC-RESULT-NL (1, P2) indication TC-RESULT-L (1, P3) TC reçoit RR-L (1, P3) duplication de P3: indication TC-L-REJECT (1, code de problème)	 indication TC-INVOKE (1, essai) demande TC-RESULT-NL (1, P1) demande TC-RESULT-NL (1, P2) demande TC-RESULT-L (1, P3)
	indication TC-R-REJECT (1, code de problème)
temps	

2.4.3 Arrivée hors séquence de composants

L'ordre des résultats segmentés n'est pas significatif pour le TC. S'il l'est, il appartiendra au protocole d'application de fournir les mécanismes nécessaires (par exemple en introduisant un mécanisme de numérotation dans les réponses intermédiaires afin de les identifier, ou encore en utilisant un réseau en mode connexion).

Du fait d'une arrivée hors séquence, un résultat partiel peut arriver après un résultat final. Lorsque cela se produit, le résultat partiel est rejeté par le TC. Cela est dû au fait que le résultat final amène le TC à fermer l'automate de lancement associé à cette opération; lorsque le résultat partiel en retard est reçu, il ne peut plus être alors associé à aucun automate de lancement actif.

La séquence au Tableau 9 illustre ce qui se produit dans l'exemple E1 lorsque la dernière partie du résultat est reçue avant la seconde: les deux utilisateurs du TC sont informés.

Lorsqu'un lancement d'opération corrélée est reçu après le résultat complet de l'opération à laquelle il est corrélé (à la suite d'un mauvais séquencement), l'opération correlée est rejetée.

Le TC suppose une très faible probabilité d'arrivée hors séquence; si les performances du réseau sont jugées insuffisantes en la matière, il est préférable d'utiliser un réseau en mode connexion.

2.4.4 Rejet d'un composant par le TC

Un principe général du TC est de rejeter un composant (lancement d'opération ou réponse) reçu mal formé ou hors contexte (par exemple, réception d'une réponse sans lancement d'opération préalable). Ce rejet signifie:

1) que le TC forme un composant de rejet pour informer l'initiateur que le composant est défectueux et il indique à l'utilisateur du TC local le composant de rejet qui attend d'être envoyé à l'extrémité distante, le TC fournissant toutes les informations disponibles sur la nature du composant rejeté (si l'utilisateur du TC éloigné n'a pas déjà mis fin au dialogue);

2) en réaction, l'utilisateur du TC local peut décider d'abandonner, de continuer ou de terminer le dialogue. Dans ces deux derniers cas, lorsque l'utilisateur du TC informe le TC de sa décision, l'utilisateur du TC homologue est informé du rejet.

Tableau 9/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (1, essai, classe = 1)	indication TC-INVOKE (1, essai) demande TC-RESULT-NL (1, P1)
indication TC-RESULT-NL (1, P1) indication TC-RESULT-L (1, P3) RR-NL (1, P2) PDU arrive résultat hors séquence: rejet (pas de système d'état actif) indication TC-L-REJECT (1, code de problème)	demande TC-RESULT-NL (1, P2) demande TC-RESULT-L (1, P3)
	indication TC-R-REJECT (1, code de problème)
temps	

Les cas possibles de rejet par le TC ont été présentés dans les paragraphes précédents. Le rejet par le TC a pour effet de mettre fin à l'opération à chaque fois que l'identificateur de lancement est reconnu: l'utilisateur du TC peut alors relancer l'opération terminée. Lorsque le composant rejeté n'est pas identifiable, l'utilisateur du TC local est informé du composant malformé reçu et une valeur d'identificateur de lancement NULL est incluse dans le composant de rejet qui attend d'être envoyé. Comme ce composant de rejet ne peut toutefois pas être associé à un lancement connu, la réaction appropriée pourrait être l'abandon du dialogue.

2.4.5 Expiration de la temporisation d'opération

Lorsque le TC informe l'utilisateur du TC de l'expiration de la temporisation (primitive d'indication TC-L-CANCEL), il indique que plus aucune information relative au lancement d'opération concerné ne peut être reçue (en particulier, aucun rejet). Si l'entité homologue continue d'envoyer des informations relatives à ce lancement d'opération, elles seront détruites lors de leur réception, à condition que l'identificateur de lancement de l'opération annulée n'ait pas été réattribué. La réattribution prématurée de valeurs d'identificateur de lancement est normalement évitée si les valeurs de temporisation sont correctement fixées et si la durée pendant laquelle un identificateur de lancement est "gelé" après que l'identificateur a été libéré a été déterminée. Cependant, un mécanisme, dépendant d'une réalisation, évitant la réattribution prématurée d'identificateurs de lancement est nécessaire afin de compenser les incertitudes concernant le temps de transfert d'une information, d'un utilisateur du TC à un autre, sans pour autant envisager le cas le plus défavorable (et, en général, le plus improbable).

L'indication de l'expiration de la temporisation relate une situation anormale uniquement dans le cas d'une opération de classe 1. L'utilisateur du TC est alors informé que, soit le lancement d'opération, soit la réponse a été perdue. Si aucun effet secondaire indésirable ne se produit, un autre lancement pour la même opération peut être effectué après l'expiration de la temporisation. Cela est illustré au Tablelau 10 par la séquence relative à l'exemple E1.

L'expiration de la temporisation pour une opération de classe 2 indique qu'aucun résultat négatif n'a été reçu, ni ne sera accepté pour ce lancement: elle est une indication définitive de succès (classe 2) si l'on suppose qu'il n'y a pas de possibilité de perte de message dans le réseau. Une situation similaire s'applique à la classe 3 en cas d'échec. La décision de transmettre une indication d'expiration de la temporisation pour une opération de classe 4 est prise au niveau local.

Tableau 10/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (1, essai, classe = 1)	indication TC-INVOKE (1, essai)
chute de la temporisation: indication TC-L-CANCEL (1) demande TC-INVOKE (2, essai, classe = 1)	
temps	

3 Dialogues

A chaque fois qu'une des primitives de gestion des opérations considérées au paragraphe 2 est émise, une demande est transmise au TC, mais rien n'est envoyé à l'utilisateur du TC distant jusqu'à ce qu'une primitive demandant la transmission ne soit émise. Ces primitives et leurs relations avec les primitives de gestion des opérations sont abordées dans ce paragraphe.

3.1 Groupement de composants dans un message

L'émission par un utilisateur du TC d'une primitive de gestion de composant a pour effet de construire un composant à inclure dans un message (sauf si cette primitive n'a qu'un effet local). Le message n'est transmis que lorsque l'utilisateur du TC le demande.

Il est à noter qu'un composant peut également être engendré suite à une procédure de rejet par le TC: dans ce cas, ce composant est placé dans le prochain message du dialogue, sauf si le dialogue est abandonné.

Plusieurs composants peuvent être regroupés et envoyés à l'extrémité distante dans un même message dans la mesure où la taille maximale de message n'est pas dépassée, ce qui évite la transmission de bits supplémentaires. Cela est fait sous le contrôle de l'utilisateur du TC qui spécifie explicitement quand il veut envoyer un ou plusieurs composants en instance de transmission. Ces composants sont ceux pour lesquels l'utilisateur du TC a préalablement émis une primitive de traitement avec le même identificateur de dialogue.

Tant que la couche du SCCP du SS n° 7 n'a pas assuré une capacité de segmentation et de réassemblage, l'utilisateur du TC doit veiller à ce que la taille maximale d'un message de SS n° 7 ne soit pas dépassée.

L'exemple E3 donné au Tableau 11 montre le début d'un dialogue avec un centre de service de réseau où un commutateur demande des instructions (opération 1) et reçoit une demande de connexion d'appel à une adresse de destination donnée et une demande d'envoi de l'information (par exemple, une annonce parlée ou un message à présenter) au demandeur. Les deux composants sont contenus dans un seul message.

TC-BEGIN et TC-CONTINUE sont des primitives de transmission décrites en 3.2.

Il est possible d'avoir une primitive de transmission par composant, (et par conséquent un composant au maximum dans chaque message) ou moins de primitives de transmission que de composants, ce qui permet le groupement de plusieurs composants dans un message. De plus, l'information contenue dans les paramètres des primitives de transmission (par exemple, l'information d'adressage) s'applique à tous les composants du message.

A l'extrémité d'origine, la primitive de demande de transmission vient après les primitives de gestion des composants, ce qui indique que la transmission de tous les composants précédents doit être faite immédiatement. Cela évite de lier la transmission de composants spécifiques à une primitive de transmission donnée et autorise les primitives de transmission sans aucun composant associé.

A l'extrémité de destination, la primitive indiquant la réception de composants transmis vient en premier. Elle contient l'information de commande nécessaire au TC pour remettre chacun des composants du message (s'il y en a). Le dernier composant du message est signalé à l'utilisateur du TC par le paramètre "dernier composant". Les composants sont remis à l'utilisateur du TC de destination dans l'ordre où ils ont été transmis au TC par l'utilisateur du TC d'origine.

Tableau 11/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (1, fournir instructions, classe = 1) demande TC-BEGIN (paramètres de commande)	

Tableau 11/Q.775 (*fin*)

Utilisateur du TC A	Utilisateur du TC B
	indication TC-BEGIN (paramètres de commande) indication TC-INVOKE (1, fournir instructions) demande TC-INVOKE (2, 1, connecter appel) demande TC-RESULT-L (1, envoyer info) demande TC-CONTINUE (paramètres de commande)
indication TC-CONTINUE (paramètres de commande) indication TC-INVOKE (2, 1, connecte rappel) indication TC-RESULT-L (1, envoyer info)	
temps	

3.2 Services de gestion de dialogue

Lorsque deux utilisateurs du TC coopèrent dans une application, plus d'un lancement d'opération est généralement nécessaire. Le flot de composants résultant doit être identifié de telle sorte que:

1) les composants liés au même flot puissent être identifiés;

2) les flots correspondant à plusieurs instances d'une même application puissent être identifiés et se dérouler en parallèle.

Chacun de ces flots est qualifié de dialogue par l'utilisateur du TC et est identifié par un paramètre identificateur de dialogue correspondant. Le service de gestion de dialogue assuré à cette fin est le dialogue structuré.

Quand un seul message est nécessaire pour réaliser une application répartie, le message unidirectionnel du dialogue non structuré peut être utilisé. L'initiateur n'attend pas de notification du résultat de l'opération (ne pouvant lancer que des opérations de classe 4), mais il peut recevoir notification d'une erreur de protocole, le cas échéant. La notification d'une erreur de protocole sera également acheminée dans un message unidirectionnel.

3.2.1 Dialogue structuré

3.2.1.1 Généralités

L'utilisation de dialogues autorise la coexistence de plusieurs flots de composants indépendants entre deux utilisateurs du TC. Le paramètre identificateur de dialogue est utilisé par les primitives de gestion des opérations et par les primitives de gestion de la transmission (dialogue) afin de déterminer l'appartenance des composants aux différents dialogues.

Dans les exemples ci-après, le paramètre identificateur de dialogue est représenté (par convention) par le premier paramètre dans ces primitives, commençant par la lettre D. Chaque utilisateur du TC a sa propre référence pour un dialogue donné. Les références locales (celles qui sont utilisées à

l'interface) sont représentées ici; le mappage entre ces références locales et les références de protocoles (dénommées identificateurs de transaction) incluses dans les messages est faite par le TC.

Trois primitives ont été définies pour la gestion de dialogues dans des conditions normales. Elles indiquent l'établissement d'un dialogue (primitive TC-BEGIN), la continuation d'un dialogue (primitive TC-CONTINUE) ou la terminaison d'un dialogue (TC-END). Chacune des ces primitives peut être utilisée pour demander la transmission de 0, 1 ou plusieurs composants; ces composants peuvent contenir des informations relatives à une ou plusieurs opérations.

Le Tableau 12 illustre une séquence possible pour l'exemple E2, où la demande d'essai débute le dialogue, qui se termine lorsque le résultat de l'essai a été envoyé.

Tableau 12/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (D1, 1, essai, classe = 1) demande TC-BEGIN (D1, adresse)	
	indication TC-BEGIN (D2, adresse) indication TC-INVOKE (D2, 1, essai) demande TC-INVOKE (D2, 2, 1, sélection d'options, classe = 1) demande TC-CONTINUE (D2)
indication TC-CONTINUE (D1) indication TC-INVOKE (D1, 2, 1, sélection d'options) demande TC-RESULT-L (D1, 2, options) demande TC-CONTINUE (D1)	
	indication TC-CONTINUE (D2) indication TC-RESULT-L (D2, 2, options) demande TC-RESULT-L (D2, 1, résultat de l'essai) demande TC-END (D2)
indication TC-END (D1, normal) indication TC-RESULT-L (D1, 1, résultat de l'essai)	
temps	
NOTE – D1 et D2 sont les références locales d'un même dialogue et correspondent aux identificateurs de transactions qui apparaissent dans les messages.	

N'importe quel groupement de composants est autorisé dans les messages d'un dialogue: le TC ne vérifie pas, par exemple, qu'un message terminant un dialogue ne contient pas des lancements d'opération de classe 1.

On part du principe que l'échange des composants se fait en mode bidirectionnel: si un utilisateur du TC veut introduire certaines restrictions, par exemple un fonctionnement en mode synchrone tel que défini pour les utilisateurs de ROSE, il lui appartiendra de définir lui-même les procédures nécessaires.

3.2.1.2 Echange de messages

La transmission de messages est effectuée avec la qualité de service des services des couches sous-jacentes: aucun contrôle de flux, aucun mécanisme de correction d'erreur n'est offert par le TC.

– la première primitive de gestion d'un dialogue doit indiquer l'établissement du dialogue (TC-BEGIN). Les messages ultérieurs ne doivent pas être envoyés par l'extrémité d'origine du dialogue tant qu'un message, indiquant la continuation du dialogue, n'a pas été reçu dans la direction opposée.

– si un utilisateur du TC essaie d'envoyer un grand nombre de messages dans un court laps de temps, aucun mécanisme de contrôle de flux du TC ne l'empêchera.

– la classe 1 du SCCP, garantissant le séquencement, peut être demandée en option et indiquée par le paramètre qualité de service. Il est à noter que cette option peut ne pas être disponible de bout en bout en cas d'interfonctionnement avec un réseau qui ne la fournit pas.

3.2.1.3 Terminaison d'un dialogue

Le TC n'impose aucune restriction quant à la possibilité qu'a un utilisateur du TC de demander la terminaison d'un dialogue. Il s'ensuit que des messages peuvent être perdus si aucune précaution n'est prise par l'application concernant le moment où un dialogue peut être terminé.

En particulier, si le protocole d'application permet aux deux utilisateurs du TC d'envoyer des primitives TC-END à peu près au même moment, et si ces primitives déclenchent la transmission de composants, il est vraisemblable que certains de ces composants (sinon tous) ne seront pas remis à leur utilisateur du TC de destination respectif.

C'est à l'application de définir, si nécessaire, ses propres règles concernant le droit de terminer un dialogue: le TC ne les vérifiera pas.

N'importe quel message reçu pour un dialogue terminé est détruit s'il demande la terminaison d'un dialogue, et tout message autre qu'un message de terminaison ou d'abandon entraîne l'abandon du dialogue à l'entité distante.

Il faut noter qu'un utilisateur du TC ne peut rejeter, au moyen d'une primitive de demande TC-U-REJECT, un composant quelconque reçu dans un message END. Il est important qu'une application puisse rejeter tout composant reçu ou qu'elle soit avertie des rejets par l'extrémité distante; tous les composants doivent être alors inclus soit dans le message initial BEGIN, soit dans les messages CONTINUE suivants.

Le dialogue prend fin soit grâce à la méthode prédéterminée, soit en envoyant un message END ne contenant pas de composants ou contenant des composants REJECT (le cas échéant).

Les différences entre les trois moyens de terminer un dialogue sont décrites ci-dessous.

Terminaison prédéterminée

Une application typique est l'accès à une base de données répartie, où l'utilisateur demandeur (utilisateur du TC A) ignore où l'information qu'il cherche est localisée. L'utilisateur du TC A diffuse une demande à chaque emplacement qui pourrait avoir l'information requise et recevra, éventuellement, une réponse de l'utilisateur du TC qui détient cette information. La terminaison prédéterminée évite que des messages, en provenance des autres destinations, ne soient envoyés pour dire: "je n'ai pas cette information". Seule la destination qui répond peut, si elle le souhaite, continuer le dialogue. Par convention, toutes les autres destinations termineront le dialogue localement: lorsque l'initiateur des demandes recevra une réponse, il terminera également localement les dialogues avec les destinations qui n'ont pas répondu. Cette convention est passée entre applications: le TC ne vérifiera pas qu'elle est respectée, ni qu'elle est mentionnée dans le protocole TC.

L'exemple E4 dans le Tableau 13 illustre cette situation dans les conditions suivantes: deux destinations B1 et B2; deux dialogues (D1, D2) et (D3, D4) sont engagés; B1 possède l'information demandée et décide de continuer le dialogue.

Une terminaison prédéterminée peut également être utilisée lorsque l'utilisateur du TC veut envoyer l'information sans attendre de réponse d'aucune sorte ultérieurement.

Tableau 13/Q.775

Utilisateur du TC A	Utilisateur du TC B1	Utilisateur du TC B2
demande TC-INVOKE (D1, 1, question) demande TC-BEGIN (D1, adresse) demande TC-INVOKE (D3, 1, question) demande TC-BEGIN (D3, adresse)	indication TC-BEGIN (D2, adresse) indication TC-INVOKE (D2, 1, question) demande TC-RESULT-L (D2, 1, réponse) demande TC-CONTINUE (D2)	indication TC-BEGIN (D4, adresse) indication TC-INVOKE (D4, 1, question) B2 ne détient pas l'information recherchée: demande TC-END (D4, local)
indication TC-CONTINUE (D1) indication TC-RESULT-L (D1, 1, réponse) D1 continue D3 se termine localement demande TC-END (D3, local)		
temps		

Terminaison de base

La réception d'une primitive de demande TC-END d'un utilisateur du TC provoque la transmission de composants en suspens à l'extrémité distante. Le TC ne vérifie pas que tous les lancements d'opération ont reçu une réponse lorsque la terminaison du dialogue est demandée: aucune information n'est donnée à l'utilisateur du TC que des lancements d'opération n'ont pas reçu de résultat complet.

A l'extrémité de réception, le dialogue est considéré comme terminé lorsque tous les composants reçus, dans un message indiquant la terminaison, ont été remis à l'utilisateur du TC.

Exemple: le dialogue prend fin lorsque l'essai de l'exemple E1, Tableau 14, reçoit une réponse.

Tableau 14/Q.775

Utilisateur du TC A	Utilisateur du TC B
.....	.....
	demande TC-RESULT-NL (D2, 1, P1) demande TC-RESULT-NL (D2, 1, P2) demande TC-RESULT-L (D2, 1, P3) demande TC-END (D2, normal) fin du dialogue pour B
indication TC-END (D1) indication TC-RESULT-NL (D1, 1, P1) indication TC-RESULT-NL (D1, 1, P2) indication TC-RESULT-L (D1, 1, P3) fin du dialogue pour A	
temps	

Abandon par l'utilisateur du TC

L'abandon permet à un utilisateur du TC d'arrêter le dialogue à n'importe quel moment. L'abandon du service par l'utilisateur est un cas typique d'abandon du dialogue. Les principales différences entre un abandon et une terminaison normale sont les suivantes:

– les composants dont la transmission est en suspens ne sont pas envoyés à l'entité homologue;

– l'information utilisateur du TC entre entités homologues peut être transmise lorsque l'abandon est émis et remise à l'utilisateur du TC distant.

La séquence donnée au Tableau 15 montre un abandon par l'utilisateur dans l'exemple E2.

Tableau 15/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (D1, 1, essai, classe = 1) demande TC-BEGIN (D1, adresse)	
	indication TC-BEGIN (D2, adresse) indication TC-INVOKE (D2, 1, essai) demande TC-INVOKE (D2, 2, 1, sélection des options, classe = 1) demande TC-CONTINUE (D2)
indication TC-CONTINUE (D1) indication TC-INVOKE (D1, 2, 1, sélection d'options) demande TC-U-ABORT (D1, cause)	
	indication TC-U-ABORT (D2, cause)
temps	

3.2.1.4 Situations anormales relatives aux messages

Ces situations sont considérées indépendamment des effets de ces événements dans la sous-couche composant.

Perte de message

Le TC n'offre aucune protection contre la perte de message. Trois cas sont identifiés:

1) le message débute un nouveau dialogue: le dialogue n'existera qu'à l'extrémité d'origine, et aucun message ne sera accepté quelle que soit la direction. Eventuellement, un mécanisme, dépendant d'une réalisation à l'extrémité d'origine, terminera le dialogue.

2) le message continue un dialogue existant: la perte n'est pas détectée. Le TC réagira (ou non) à la perte des composants contenus dans le message comme indiqué en 2.4.1.

3) le message termine un dialogue: le TC réagira éventuellement (par l'expiration de la temporisation comme indiqué en 2.4.1) si ce message contient une réponse à une opération de classe 1, sinon un mécanisme dépendant d'une réalisation peut terminer le dialogue à l'extrémité de destination.

Duplication de message

La duplication d'un message BEGIN provoque l'ouverture de deux transactions comme indiqué ci-dessous: chacune de ces transactions a son propre identificateur local et le même identificateur de

destination. L'utilisateur du TC détecte éventuellement une situation anormale, et les deux dialogues sont abandonnés.

La séquence donnée au Tableau 16, illustre la duplication d'un message BEGIN dans l'exemple E2.

Tableau 16/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (D1, 1, essai, classe = 1) demande TC-BEGIN (D1, adresse)	
indication TC-CONTINUE (D1) indication TC-INVOKE (D1, 2, 1, sélection d'options)	indication TC-BEGIN (D2, adresse) indication TC-INVOKE (D2, 1, essai) duplication de BEGIN: indication TC-BEGIN (D3, adresse) indication TC-INVOKE (D3, 1, essai) réponse au premier Begin demande TC-INVOKE (D2, 2, 1, sélection d'options, classe = 1) demande TC-CONTINUE (D2) réponse au second Begin demande TC-INVOKE (D3, 2, 1, sélection d'options, classe = 1) demande TC-CONTINUE (D3)
indication TC-CONTINUE (D1) indication TC-INVOKE (D1, 2, 1, sélection d'options) l'utilisateur du TC considère que ce lancement est anormal et peut le rejeter ou abandonner un des dialogues: demande TC-U-ABORT (D1, cause)	
	indication TC-U-ABORT (D3, cause)
temps	

A cet instant, il y a encore un dialogue (avec l'identificateur local D2) à l'extrémité B, mais aucun dialogue à l'extrémité A. L'utilisateur du TC B recevra une indication du TC lorsque la temporisation de l'opération 2 du dialogue D2 expirera faute de réponse (primitive d'indication TC-L-CANCEL), et pourra alors décider d'abandonner D2. Il est à noter que la situation serait plus difficile à détecter si l'utilisateur du TC B n'avait pas lancé une opération de classe 1.

La duplication d'un message CONTINUE n'est pas détectée par le TC.

Lorsqu'un messsage END est dupliqué, le second message est reçu avec un identificateur qui ne correspond pas à un dialogue en cours: le TC détruit le message dupliqué.

Arrivée hors séquence de messages

Lorsque les messages hors séquence ne concernent ni l'établissement, ni la terminaison d'un dialogue, le mauvais séquencement n'est pas détecté par le TC mais peut entraîner un mauvais séquencement des composants, auquel le TC réagit comme indiqué en 2.4.3.

Lorsqu'un message indiquant la continuation d'un dialogue arrive après un message indiquant la terminaison de ce même dialogue, il n'est pas remis et provoque l'abandon du dialogue par le TC; l'utilisateur du TC détectera probablement la perte en recevant une indication de terminaison de dialogue prématurée. Si l'application exige que ce problème soit surmonté, un nouveau dialogue doit être établi.

Message altéré

Lorsqu'un message altéré est reçu, le TC réagit comme indiqué dans la Recommandation Q.774.

La séquence de primitives, lorsque le TC décide d'abandonner le dialogue suite à la réception d'un message altéré dans l'exemple E2, est indiquée dans le Tableau 17.

Tableau 17/Q.775

Utilisateur du TC A	Utilisateur du TC B
demande TC-INVOKE (D1, 1, essai, classe = 1) demande TC-BEGIN (D1, adresse)	
	indication TC-BEGIN (D2, adresse) indication TC-INVOKE (D2, 1, essai) demande TC-INVOKE (D2, 2, 1, sélection d'options, classe = 1) demande TC-CONTINUE (D2)
message altéré: indication TC-P-ABORT (D1, cause)	indication TC-P-ABORT (D2, cause)
temps	

3.2.1.5 Relations entre la gestion du dialogue et la gestion des composants

La suite donne quelques directives sur le moment où la terminaison d'un dialogue peut être demandée; si ces directives ne sont pas respectées, le TC ne refusera pas la demande de terminaison du dialogue.

Les problèmes qui peuvent résulter de la collision de messages demandant la terminaison du dialogue ont été décrits précédemment.

Une terminaison normale ne devrait pas être demandée lorsque:

– il y a des lancements d'opération en cours pour le dialogue;

– le protocole d'application prévoit que les réponses transmises avec la demande de terminaison pourraient être rejetées.

De nombreuses applications pourraient ne pas définir les scénarios de rétablissement en cas de rejet d'une réponse. Cela justifie la transmission de réponses ou d'opérations de classe 4 dans un message indiquant la terminaison d'un dialogue. Les autres applications devraient, soit envisager l'utilisation d'un service réseau en mode connexion, soit terminer le dialogue par un message ne contenant pas de composant, qui serait envoyé seulement lorsqu'une indication de rejet ne peut plus être reçue.

Il est recommandé de ne pas envoyer une opération pour laquelle une réponse est attendue dans un message END (c'est-à-dire que l'utilisateur du TC n'émette pas de primitive de demande TC-END pour déclencher le composant associé). Il convient de faire observer qu'il ne s'agit pas d'une condition du protocole du TC ni de l'utilisateur du TC mais d'une directive judicieuse. Si l'utilisateur du TC choisit d'envoyer un composant de lancement pour une opération de classe 1, 2 ou 3 dans un message END, son choix n'a aucun effet perturbeur pour les systèmes de protocole du TC homologues ni pour les utilisateurs de TC. La manière dont les composants produits à la suite de l'exécution de l'opération en l'absence de dialogue actif sont détruits est à déterminer au niveau local. A vrai dire, le fait que l'utilisateur du TC qui a lancé l'opération ait choisi de les inclure dans un message END signifie qu'il a décidé pour cet exemple de passer outre à la sémantique inhérente à l'opération et qu'il ne se soucie guère de savoir si l'opération a pu ou non être exécutée.

3.2.1.6 Questions d’adressage

L'utilisateur du TC qui débute un dialogue doit donner l'adresse de destination et l'adresse d'origine au TC. Il appartient à l'utilisateur du TC qui fournit l'adresse d'origine de la donner de manière que le TC distant puisse l'utiliser, sans autre vérification, pour atteindre cette destination.

Quand un SCCP sans connexion assurant les capacités du TC est utilisé, toutes les options d'adressage qu'il offre peuvent être choisies. Toutes les combinaisons de types d'adresses de destination et d'origine sont donc autorisées.

Pendant l'instauration d'un dialogue, la combinaison des adresses peut être optimisée à la fois par l'utilisateur du TC B et par le TC.

Le paramètre facultatif d'adresse d'origine dans la première primitive de demande TC-CONTINUE peut servir à modifier l'adresse qui devrait être utilisée pour acheminer les messages ultérieurs associés au dialogue. La destination effective ne devrait pas être affectée par cette modification. S'il s'agit d'une nouvelle destination, il faudra mettre fin au dialogue et en commencer un nouveau vers la nouvelle destination.

Exemples de modification d'adresse qui n'affectent pas la destination effective:

– l'adresse initiale était un titre global, l'adresse suivante est un PC et un SSN pour la même destination afin de permettre un acheminement optimal (qui ne traverse pas la limite du réseau);
– un titre global général est utilisé pour sélectionner une base de données à partir d'un ensemble de bases de données répétées. La base de données interrogée renvoie son propre titre global spécifique.

3.2.1.7 Qualité de service

Les primitives de demande de gestion du dialogue permettent à l'utilisateur du TC d'exiger une qualité de service donnée de la couche Réseau. Si l'utilisateur du TC ne formule aucune demande particulière, le SCCP sélectionne des options par défaut (c'est-à-dire pas d'option retour, pas de mise en séquence).

Si le contrôle de séquence est demandé, l'utilisateur du TC est aussi chargé de fournir les informations permettant à la couche Réseau d'identifier le flux des messages connexes qui doivent être remis en séquence.

Il est conseillé aux utilisateurs de TC de demander le contrôle de séquence lorsqu'ils utilisent les services TC-RESULT-NL.

Lorsque l'option retour est demandée, l'utilisateur du TC est informé de la non-remise du message associé par la primitive d'indication TC-NOTICE.

Bien que la primitive d'indication TC-NOTICE soit essentiellement destinée à être traitée par une fonction de gestion (par exemple, lorsqu'elle indique qu'une défaillance s'est produite pendant la traduction d'un titre global), il y a des cas dans lesquels l'option retour peut être demandée dans une primitive de demande TC-BEGIN ou de demande TC-UNI pour déterminer s'il existe une entité d'application de destination (identifiée par le numéro du sous-système dans l'adresse de destination) dans l'entité de réception. En pareil cas, la primitive d'indication TC-NOTICE pourra être interprétée en temps réel.

Dans l'exemple qui suit, l'utilisateur du TC A qui se trouve dans un commutateur local lance une procédure de préanalyse avant d'établir une communication. Il demande l'option retour au moment où il émet la primitive d'instauration de dialogue TC-BEGIN en direction du commutateur local B de destination. La procédure de préanalyse n'est pas mise en œuvre dans le nœud B. Il n'existe donc pas de numéro de sous-système correspondant à la procédure de préanalyse et le SCCP dans le nœud B renvoie le message SCCP contenant le message BEGIN. L'utilisateur du TC A reçoit un message de confirmation par l'intermédiaire de la primitive d'indication TC-NOTICE et il engage une procédure d'établissement (de circuit) d'appel de base.

Tableau 18/Q.775

Utilisateur du TC A		**Utilisateur du TC B**
lancement du traitement d'appel demande TC-INVOKE (D1, 1, préanalyse (LookAhead)) demande TC-BEGIN (D1, option retour, destination = adresse B, origine = adresse A)		
	UNITDATA (option retour, adresse B, adresse A, BEGIN)	
		le numéro SSN-B n'existe pas
	UNITDATA SERVICE (cause du retour, adresse A, adresse B, BEGIN)	
indication N-NOTICE (D1, cause du retour, destination = adresse A, origine = adresse B)		

3.2.2 Dialogue non structuré

Un message unidirectionnel ne contient que des invocations d'opération de classe 4, ou des notifications d'erreurs de protocole pour ces invocations. Des composants multiples peuvent être transmis dans un message unidirectionnel, à condition que la longueur maximale du message ne soit pas dépassée.

3.3 Fonctionnalités améliorées de gestion du dialogue

3.3.1 Aperçu général

Etant donné que le nombre d'applications de signalisation utilisant le TC augmente, il faudra pouvoir les différencier pendant une communication, notamment lorsque plusieurs d'entre elles se trouvent au même endroit dans un nœud du SS n° 7.

La possibilité de signaler au début du dialogue le protocole d'application (parmi plusieurs éventuellement) qui intervient dans l'échange ultérieur de messages est assurée par les fonctions facultatives et le protocole de portion de dialogue. La portion de dialogue facultative permet la négociation du contexte d'application et, à titre d'option supplémentaire, le transfert transparent des données d'utilisateur qui ne sont pas des composants. Ces dernières pourraient servir par exemple à acheminer des données d'initialisation, des versions de protocoles d'utilisateur, d'autres précisions du contexte d'application, des mots de passe, etc.

3.3.2 Utilisation du contexte d'application

L'utilisateur du TC qui commence un dialogue peut proposer un contexte d'application à son homologue en incluant un nom de contexte d'application dans la primitive de demande TC-BEGIN. Le contexte d'application se rapporte à l'ensemble d'éléments ASE et aux règles de coordination associées qui peuvent être nécessaires pendant le dialogue.

Si le nom du contexte d'application est acceptable, l'utilisateur du TC qui répond peut soit décider de poursuivre le dialogue, soit y mettre fin normalement. Il doit inclure le même nom de contexte d'application dans la première (ou l'unique) primitive de demande de gestion du dialogue qu'il utilise sauf s'il demande une terminaison de dialogue prédéterminée.

Si le nom du contexte d'application n'est pas acceptable, l'utilisateur du TC peut choisir:

i) de détruire les composants reçus et d'émettre une primitive de demande TC-U-ABORT pour faire savoir qu'il refuse le dialogue. Il doit inclure dans cette primitive un nom de contexte d'application qui correspond soit à celui qui a été reçu, soit à un autre qui sera utilisé par l'initiateur du dialogue pour faire une nouvelle tentative. Le TC n'a pas de fonction type permettant à son utilisateur de proposer plus d'un nom de contexte d'application de rechange. Une procédure de ce type peut toutefois être définie en dehors du TC avec le paramètre d'information d'usager (voir 3.3.3);

ii) de poursuivre le dialogue en indiquant que celui-ci utilise un autre contexte d'application [par exemple un contexte qui n'utilise pas le(s) ASE(s) qu'il ne prend pas en charge] en incluant un autre nom de contexte dans la première primitive de demande TC-CONTINUE. Les composants reçus peuvent être conservés ou détruits selon l'accord préalablement conclu entre les utilisateurs de TC;

iii) de mettre fin au dialogue normalement en indiquant que la (les) réponse(s) qui figure(nt) dans le message END repose(nt) sur un autre contexte d'application [par exemple un contexte qui n'utilise pas l'(les) ASE qu'il ne prend pas en charge] en incluant un autre nom de contexte d'application dans la primitive de demande TC-END.

L'utilisateur du TC peut aussi donner un nom de contexte d'application lorsqu'il se sert d'un service UNI-TC. Dans ce cas, le nom du contexte permet à l'utilisateur du TC homologue d'interpréter les composants reçus.

Il importe de noter que l'information relative au contexte d'application qui est acheminée dans l'APDU de gestion du dialogue est un nom du type OBJECT IDENTIFIER (IDENTIFICATEUR D'OBJET). Ce nom fait référence à une spécification (document) dans laquelle le contexte d'application est décrit. Ce document peut renvoyer à d'autres spécifications où, par exemple, la syntaxe abstraite d'un protocole d'application donné est décrite. Des spécifications de ce type peuvent être énoncées sous forme de notation formelle ou semi-formelle ou encore de texte clair.

La spécification des contextes d'application, leur sémantique, le choix d'une valeur d'identificateur d'objet et la diffusion de l'information à toutes les parties qui souhaitent communiquer sont autant de modalités qui permettent d'enregistrer un contexte d'application. Dans le cas où les contextes d'application sont enregistrés dans des Recommandations relatives à la signalisation du RNIS, un exemple de valeur type pourrait être {ccitt recommendation q xxx ac-name (y)} pour le contexte d'application y décrit dans la Recommandation q.xxx.

S'il est en principe possible de décrire le contexte d'application en détail et d'ajouter de nouveaux contextes d'application pour tenir compte de tous les cas qui peuvent se produire, il est sans doute plus facile de conserver ces spécifications si les contextes d'application ne sont pas trop nombreux. Supposons par exemple qu'un nom de contexte d'application donné se réfère à l'utilisation combinée des ASE A et B. Dans certains cas, il peut être nécessaire d'indiquer que seul un sous-ensemble des fonctions de A et/ou B sera utilisé pour une communication donnée. Au lieu d'enregistrer un nouveau nom de contexte d'application pour couvrir ce cas, la même information peut être acheminée dans une syntaxe acceptable par les deux parties dans le champ d'information d'usager des APDU AARQ et AARE.

3.3.3 Transfert des données d'utilisateur

Les primitives de gestion du dialogue du TC permettent à l'utilisateur du TC de transférer des informations qui n'ont pas trait aux services de gestion des composants (c'est-à-dire qui ne reposent pas sur le paradigme de l'opération distante). Ces informations sont acheminées soit dans le champ d'information-utilisateur de la PDU de gestion du dialogue, soit directement dans la portion de dialogue une fois ce dernier établi.

Un service de ce type est nécessaire au moment de l'établissement du dialogue pour transmettre certaines données d'initialisation à l'homologue (précisions relatives au contexte d'application, données d'authentification, identification d'un sous-processus de destination, etc.)

Ce service peut aussi être utilisé pour négocier le contexte d'application quand l'utilisateur du TC refuse un dialogue (abandon par l'utilisateur pendant le dialogue avec une raison pour l'abandon, à savoir l'absence de prise en charge du contexte d'application), il peut insérer une liste de noms de contextes d'application de rechange dans le champ de données d'utilisateur de la primitive de demande TC-U-ABORT. Ces noms sont alors acheminés dans le cadre des données d'utilisateur de l'unité de données de protocole de dialogue (ABRT). L'utilisateur du TC qui est à l'origine de la demande d'établissement du dialogue peut faire une nouvelle tentative avec l'un de ces contextes.

Pour utiliser ce service, les deux utilisateurs de TC devront définir la syntaxe et la sémantique de l'information à acheminer, s'il y en a une, dans chaque APDU de dialogue pour tous les contextes d'application. Comme le type de cette information d'utilisateur en ASN.1 est EXTERNAL, la syntaxe de l'information peut être écrite en notation ASN.1 ou en toute autre notation spécifique à l'utilisateur. La manière dont l'information est codée peut aussi être spécifique à l'utilisateur. Le type EXTERNAL permet d'intégrer la valeur de données à partir d'une syntaxe abstraite (dans ce cas une syntaxe spécifique à l'utilisateur) dans une autre (celle des APDU de dialogue).

La Recommandation X.208 définit le type EXTERNAL comme suit:

```
EXTERNAL ::= [UNIVERSAL 8] IMPLICIT SEQUENCE {
    direct-reference            OBJECT IDENTIFIER OPTIONAL,
    indirect-reference          INTEGER OPTIONAL,
    data-value-descriptor       ObjectDescriptor OPTIONAL,
    encoding                        CHOICE {
        single-ASN1-type                [0] ANY,
        octet-aligned                   [1] IMPLICIT OCTET STRING,
        arbitrary                       [2] IMPLICIT BIT STRING }}
```

Parmi les trois formes de dénotation qui permettent d'identifier le type et le codage de la valeur de données qui est contenue dans la construction EXTERNAL, les utilisateurs de TC doivent utiliser la dénotation directe. Le nom de la dénotation directe est essentiel pour identifier à la fois la syntaxe abstraite de la valeur de données et les règles de codage qui s'appliquent. La dénotation indirecte sert à identifier le contexte de présentation dont l'utilisation n'est pas, actuellement, prise en charge par le réseau de signalisation n° 7. En plus de la dénotation directe, l'utilisateur du TC peut aussi décrire expressément la valeur de données dans une notation informelle en ayant recours au descripteur de la valeur de données.

Si la valeur de données externe correspond à un type d'ASN.1 unique et si les règles de codage de base servent à coder cette valeur, le choix du champ de "codage" importe peu. Si le codage de cette valeur de données qui a été choisi se traduit par un nombre entier d'octets, le codage "aligné-octets" ou "arbitraire" peut être utilisé. Dans le cas contraire, il convient de choisir l'option "arbitraire".

Comme le protocole admet la présence d'une SEQUENCE OF EXTERNAL dans un champ d'information d'utilisateur facultatif des APDU de gestion du dialogue, les deux utilisateurs de TC ne sont pas limités, lorsqu'ils définissent un contexte d'application, à un nombre donné dans la séquence. (Si la segmentation n'est pas assurée par le SCCP, les utilisateurs de TC devront veiller à ce que la taille des messages du système de signalisation n° 7, qui est limitée, soit respectée).

3.3.4 Questions de compatibilité amont

Les fonctions et le protocole nouveaux décrits dans les sous-paragraphes ci-dessus sont facultatifs et les procédures de protocole qui sont spécifiées dans les Recommandations Q.771 à Q.774 sont faciles à distinguer et à retirer de la documentation remise à l'utilisateur, des spécifications relatives aux mises en œuvre et à l'interface entre les réseaux qui prennent ces Recommandations comme base. Dans ce cas, il convient de se reporter aux messages du TC définis dans les Recommandations de 1988. Aucun réseau n'est obligé d'accepter ces nouvelles fonctions s'il n'offre pas des services qui les exigent.

Un nœud qui accepte la version du TC de 1988 ne comprendra pas les APDU associées au contexte d'application dû à un nœud conforme à la Recommandation de 1992 (ou à toute version ultérieure) et arrêtera donc la transaction en utilisant la cause d'abandon partiel "portion de transaction incorrecte". Si l'utilisateur du TC à un nœud qui prend en charge le TC conforme à la Recommandation de 1992 (ou à toute version ultérieure) reçoit un message d'abandon avec la cause susmentionnée en réponse à un début de dialogue utilisant l'information du contexte d'application, il doit l'interpréter, du moins dans les cas où un réseau prend en charge diverses mises en œuvre du TC d'après la Recommandation de 1992 (ou à toute version ultérieure) et les Recommandations de 1988 parce qu'il communique avec un nœud qui n'accepte que les Recommandations relatives au TC de 1988 et non en raison d'une véritable erreur de syntaxe (qui est très peu probable). L'utilisateur du TC peut donc essayer de retransmettre le message sans autres informations supplémentaires relatives au contexte d'application que celles qui sont essentielles à l'application.

Il est vraisemblable que pendant un certain temps, un réseau acceptera les mises en œuvre du TC conformes aux Recommandations de 1988 et à la Recommandation de 1992 (ou à toute version

ultérieure) ainsi que des applications qui exigent ou non la prise en charge du mécanisme de contexte d'application. Ces possibilités ne relèvent pas des présentes normes mais ne doivent pas être oubliées au moment où les services sont déployés si l'on veut éviter de transmettre le message initial deux fois.

4 Directives pour l'élaboration des protocoles utilisateurs de TC

4.1 Introduction

La Recommandation Q.1400 décrit la manière dont les éléments de service d'application (ASE, *application service element*), les contextes d'application (AC, *application-context*) et les entités d'application (AE, *application entity*) sont structurés et celle dont une AE est traitée dans le système de signalisation n° 7. Le présent paragraphe illustre cette architecture en tenant compte de la découpe fonctionnelle d'une application et décrit la manière dont les AE, les AC, les ASE, les opérations et les erreurs devraient être définis.

4.2 Découpe fonctionnelle dans une application

4.2.1 Processus d'application et entité d'application

Un processus d'application (AP, *application process*) de signalisation communique à travers une partie de son logiciel consacré exclusivement aux communications qui est appelée entité d'application. Une AE contient donc toutes les fonctions nécessaires pour que les divers AP puissent communiquer. Pour résumer ce qui est dit au sous-paragraphe 4.1/Q.1400, un type d'entité AE est un ensemble de protocoles de communication spécifique à une application. Le type d'entité AE est à définir au niveau local. Une entité AE est la réalisation du type d'entité AE correspondant dans une entité physique.

4.2.2 Elément de service d'application

A maintes reprises, on a constaté que pour un certain nombre d'applications, les fonctions de communication pouvaient être groupées dans des ensembles intégrés d'actions de manière que chacun de ces ensembles puisse être utilisé dans plusieurs AE. Un ensemble intégré d'actions de ce type qui peut être utilisé dans plusieurs AE est appelé un élément de service d'application (ASE, *application service element*). Il va sans dire qu'il existe toujours certaines fonctions de communication spécifiques à une application qui ne peuvent servir qu'à répondre aux besoins de l'application donnée pour laquelle elles ont été définies.

Le TC offre un moyen général à toutes les applications de signalisation de communiquer en utilisant le paradigme des opérations distantes sur un service de réseau sans connexion.

Un ASE utilisateur du TC est constitué d'opérations distantes qui, ensemble avec le TC, fournissent un protocole global de communication à une application de signalisation. La définition de l'ASE indique aussi quel utilisateur du TC homoloque peut lancer une opération, en précisant la nature de cette dernière, et dans quel ordre. Si l'un ou l'autre des utilisateurs du TC peut lancer n'importe quelle opération, on dit que l'ASE est symétrique. La manière dont les opérations sont définies et groupées est décrite en 4.5.

Du point de vue de l'utilisateur du TC, le mécanisme qui permet d'obtenir les services d'un ASE utilisateur du TC est le lancement des opérations de ce dernier. Chaque opération fournit une partie du service de l'ASE de manière intrinsèquement asymétrique car elle est lancée par un utilisateur du TC et exécutée par l'homologue distant. Les utilisateurs de TC ne sont toutefois pas toujours asymétriques (c'est-à-dire qu'un utilisateur ne se limite pas toujours à exécuter des opérations et un autre à les lancer) car chacun peut être en mesure de lancer et d'exécuter les mêmes opérations ou des

opérations différentes. [En fait, l'interface du service du point de vue de l'utilisateur du TC (qui fera l'objet d'un complément d'étude) peut sembler très différente de celle qui est fournie par le TC. Le lancement par exemple d'une opération de classe 1 peut être considéré par l'utilisateur du TC comme le lancement d'un service confirmé alors que dans la perspective de l'interface du service du TC, il résulte de deux services non confirmés, à savoir TC-INVOKE et TC-RESULT].

Certains contextes d'application peuvent nécessiter l'utilisation d'un ASE utilisateur du TC spécifique pour établir un dialogue et y mettre fin. Cet ASE tient compte de deux opérations spécifiques (appelées opérations de rattachement et de détachement) qui forment un lot de connexion.

D'autres applications peuvent aussi nécessiter l'utilisation d'autres ASE non fondés sur l'utilisation d'opérations. Les unités PDU de ces ASE sont acheminées dans la portion de dialogue.

4.2.3 Communications entre AE/ASE homologues

Deux entités AE situées dans des nœuds différents peuvent communiquer entre elles à condition qu'elles connaissent l'une et l'autre l'adresse de leur AE homologue et qu'elles acceptent un contexte d'application commun. Un contexte d'application spécifie les règles, les procédures et le comportement à observer à l'interface externe entre deux AE en communication. Les modalités de spécification d'un contexte d'application sont définies au 4.3.

Pour les applications utilisant le TC conformes aux Recommandations de 1988 (*Livre bleu*), faisant appel au service SCCP UNITDATA, le numéro de sous-système (SSN) du SCCP assure l'acheminement du message du système de signalisation n° 7 jusqu'à l'AE dans le cadre d'une entité physique acceptant le protocole d'application précis utilisé ainsi que, implicitement, le codage utilisé pour ses messages. En d'autres termes, outre qu'il constitue un élément d'information d'adressage, le SSN définit également implicitement le contexte d'application (en gros, l'ensemble de messages) et le contexte de présentation (le codage des messages).

Avec l'introduction du protocole pour la portion de dialogue du TC, qui contient le mécanisme de négociation du contexte d'application, il n'existe aucune relation directe entre l'entité AE et le numéro SSN. Le contexte d'application offre un moyen explicite de reconnaître la granularité dans la couche d'Application. Etant donné qu'une entité AE peut prendre en charge n'importe quel nombre d'éléments ASE (voir 6.1.3/Q.1400), le contexte d'application identifie l'ensemble approprié d'éléments ASE à utiliser pendant le dialogue SS n° 7.

Ces points sont illustrés sur les figures ci-dessous.

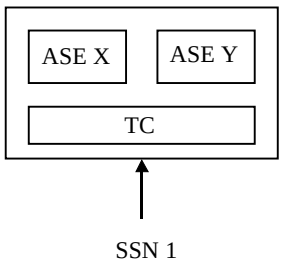


Figure A

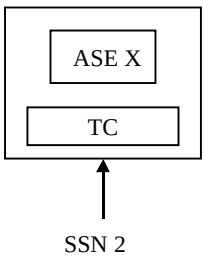


Figure B

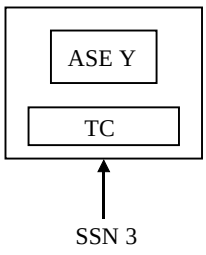


Figure C

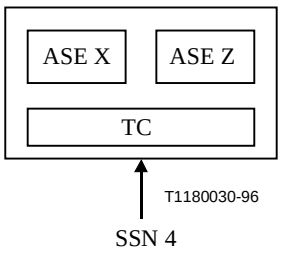


Figure D

Chacun des quatre grands rectangles représente un type d'entité AE, comportant plusieurs éléments ASE dont un, dans le cas qui nous intéresse ici, est toujours le TC. Les éléments ASE X, Y et Z représentent divers éléments ASE utilisateurs du TC, tels que ceux à utiliser pour les services complémentaires. Chaque type d'entité AE est accessible par un numéro SSN distinct.

Les Figures A et B montrent deux mises en œuvre locales possibles de l'élément ASE X. Dans un cas (Figure A) l'élément ASE X a été placé avec l'élément ASE Y dans un type d'entité AE accessible

par un numéro SSN donné (éventuellement choisi au niveau local). Dans la Figure B, l'élément ASE X a été placé isolément. Les services assurés par l'élément ASE X admettent deux mises en œuvre possibles dont le choix n'est pas soumis à normalisation. Dans la Figure D, par exemple, on a retenu une autre mise en œuvre regroupant l'élément ASE X et un élément ASE complètement différent, l'élément ASE Z, pour former un type d'entité AE accessible par le numéro SSN 4.

Il est à noter que la diversité des types d'entité AE représentés sur les figures ci-dessus a pour but de souligner que les types d'entités AE *ne sont pas* normalisés, chaque nœud n'ayant donc pas à choisir la même mise en œuvre. Les communications à établir pour assurer les services de l'élément ASE X demeurent possibles entre deux mises en œuvre différentes, telles que représentées par exemple sur la Figure A et sur la Figure B ou D, à condition que les données d'acheminement SCCP soient suffisamment complètes (ce qui relève d'un problème d'administration et de gestion du réseau) et que les opérations/paramètres/codes d'erreurs de l'élément ASE X diffèrent de ceux des éléments ASE Y et Z (ce qui est à déterminer au moment de la spécification; voir 4.5.8 pour de plus amples précisions).

Le seul point faisant l'objet d'une normalisation est le comportement externe d'un système. Les contextes d'application définissent le comportement externe visible entre deux entités AE en communication. Certains contextes AC peuvent être normalisés. Les éléments ASE sont souvent normalisés, ce qui permet de les réutiliser dans de nombreux contextes d'application différents.

4.3 Comment spécifier un contexte d'application

Au cours d'une instance de communication, les interactions entre deux entités AE ainsi que les interactions entre les éléments ASE dans une AE sont régies par les règles d'un contexte d'application (AC).

La définition d'un AC doit au moins contenir:

– une description générale;

– une définition du protocole d'application complet entre AE homologues en:

  i) identifiant chaque ASE utilisé par l'AC et en indiquant quelles AE homologues initient le service;

  ii) indiquant les règles de coordination utilisées entre ces ASE (par exemple concaténation des PDU d'ASE utilisateurs du TC différents, contraintes relatives à l'ordre dans lequel les opérations provenant d'ASE utilisateurs du TC différents peuvent être lancées, etc.) outre les règles qui font partie intégrante des spécifications relatives aux ASE;

  iii) précisant la (les) syntaxe(s) abstraite(s) requise(s) par les ASE;

– les contraintes spéciales visent à garantir la compatibilité entre différentes versions d'AE homologues.

Un nom doit être donné à chaque contexte d'application. Ce nom est une valeur du type OBJECT IDENTIFIER (IDENTIFICATEUR D'OBJET) qui est acheminée (le cas échéant) en tant que valeur de l'élément d'information nom du contexte d'application dans la portion de dialogue.

Une classe d'objet d'information ASN.1 pouvant servir à définir les aspects statiques des contextes d'application utilisant le TC est spécifiée au sous-paragraphe 5.4.

4.4 Comment spécifier un ASE

La spécification d'un ASE utilisateur du TC doit fournir au moins:

– une description générale de l'ASE et de ses procédures;

– la liste des opérations assurées ainsi que l'indication de l'extrémité (ou les deux) qui peut lancer les opérations en précisant leur nature;
– les règles sur la séquence dans laquelle les opérations peuvent être lancées;
– la description précise des procédures;
– les modalités d'interfonctionnement des différentes versions de protocole;
– la description des interactions entre l'ASE et le TC du point de vue des primitives du service du TC;
– les diagrammes SDL.

Les classes d'objet d'information OPERATION-PACKAGE et CONNECT-PACKAGE définies dans la Recommandation X.880 peuvent être utilisées pour spécifier les aspects statiques de la définition des ASE basés sur l'utilisation d'opérations.

4.5 Comment spécifier les opérations et les erreurs

4.5.1 Généralités

L'ensemble des opérations et des erreurs qui constitue une spécification du protocole d'utilisateur du TC peut être décrit à l'aide d'un ou de plusieurs module(s) ASN.1. Le nombre de modules ASN.1 à utiliser est laissé à l'appréciation du concepteur de protocole et cette question est examinée plus en détail en 4.5.5.

Une notation pouvant permettre de définir les opérations et les erreurs est basée sur le service MACRO ASN.1 défini dans la Recommandation X.208. OPERATION MACRO et ERROR MACRO sont les types de données associés respectivement à une opération et à une erreur. Toutefois, la notation MACRO étant en voie d'abandon en ASN.1, la Recommandation X.880 spécifie une autre notation utilisant deux classes d'objet d'information équivalentes, expliquant, entre autres, comment on peut passer de la notation MACRO à la notation de classe d'objet.

Chaque opération (erreur) appartient à un type d'opération (type d'erreur) qui est dérivé du type OPERATION MACRO (ERROR MACRO).

Il convient d'attribuer un nom (une référence de type ASN.1 qui commence par une lettre majuscule) à chaque type d'opération ou type d'erreur.

Il convient d'attribuer un nom (une référence de valeur ASN.1 qui commence par une lettre minuscule) à chaque opération ou erreur.

La Recommandation X.219 précise que les valeurs d'un ensemble d'opérations et d'erreurs doivent être uniques dans une syntaxe abstraite. Dans le cas du TC, cela veut dire pour le moment qu'elles doivent être uniques dans le cadre d'un numéro de sous-systèmes ou d'un groupe de numéros de sous-systèmes connexes ou d'un contexte d'application.

La définition du type peut être associée à l'attribution d'une valeur ou effectuée en deux étapes, comme le montre l'exemple suivant:

– la spécification du type et l'attribution d'une valeur sont distinctes:

```
OperationTypeExample1 ::= OPERATION
ARGUMENT      ParameterType1
RESULT        ResultType1
ERRORS        { error1, error2 }

operationExample1 OperationTypeExample1 ::= localValue 1
```

– la spécification du type et l'attribution d'une valeur sont combinées:

```
OperationExample1 ::= OPERATION
ARGUMENT        ParameterType1
RESULT          ResultType1
ERRORS          { error1, error2 }
::= localValue 1
```

On trouvera en 4.5.6 les raisons pour lesquelles il est plus pratique de combiner les spécifications selon le type et selon la valeur, ainsi que des précisions sur l'utilisation de valeur globale ou locale.

La notation définie dans la Recommandation X.880 n'admet pas cette approche en deux étapes. On peut toutefois facilement modifier le code attribué à une opération ou à une erreur en utilisant l'opération paramétrée prédéfinie recode {}. Dans l'exemple qui suit, operationExample2 est défini comme étant identique à operationExample1 (même type d'opération) mais avec une valeur différente.

```
OperationExample2 ::= recode {operationExample1, 2}
```

4.5.2 Utilisation de la notation OPERATION MACRO

4.5.2.1 Utilisation de la notation de type

Un type d'opération est entièrement défini comme un exemple de type OPERATION MACRO complété par un commentaire ASN.1 indiquant la valeur de la temporisation associée.

Les sous-paragraphes ci-après donnent des directives sur l'utilisation des diverses productions ASN.1 qui constituent la description OPERATION MACRO.

4.5.2.1.1 Spécification de l'argument d'opération

Les productions ASN.1 ci-après montrent comment l'argument d'une opération doit être spécifié:

```
Parameter ::= ArgKeyword ParameterType | empty
ArgKeyword ::= "PARAMETER" | "ARGUMENT"
ParameterType ::= NamedType | NamedType "OPTIONAL"
```

Si des informations peuvent être fournies au moment du lancement de l'opération, l'un des mots clés PARAMETER ou ARGUMENT doit être inséré et suivi par le type nommé NamedType qui correspond à la structure de données à fournir, sinon le mot clé ne sera pas présent dans la description d'opération.

Les deux mots clés sont autorisés aux fins de la compatibilité amont, les spécifications relatives à l'utilisateur du TC reposant sur des versions anciennes de TC. L'utilisation du mot clé PARAMETER est toutefois déconseillée pour définir des applications nouvelles.

Bien que l'argument d'opération soit toujours un élément facultatif d'un composant de lancement Invoke, la spécification du type de paramètre ParameterType indique si la présence de cet argument au moment du lancement est obligatoire ou facultative pour la bonne exécution de l'opération. Si elle est facultative, le mot clé OPTIONAL suit le type ASN.1.

4.5.2.1.2 Spécification de résultats positifs

Les productions ASN.1 ci-après montrent comment spécifier des opérations dont les résultats sont positifs.

```
Result :: = "RESULT" ResultType | empty
ResultType :: = ParameterType | empty
```

Si l'information peut être renvoyée en tant que résultat de l'exécution d'une opération réussie, le mot clé RESULT doit être suivi du type nommé NamedType associé à la structure de données à émettre.

Si aucune information ne peut être fournie mais que la classe d'opération indique que l'opération a réussi, le mot clé RESULT doit être présent dans la description d'opération mais l'alternative vide de la production résultat est utilisée. Si le mot clé RESULT n'est pas inclus dans une description d'opération, cela indique qu'il n'y a pas de notification de réussite (c'est-à-dire opération de la classe 2 ou 4).

Bien que le paramètre de résultat d'opération soit toujours un élément facultatif d'un composant retour résultat Return Result, la spécification du type de paramètre ParameterType indique si la présence de ce paramètre est obligatoire ou facultative d'un point de vue fonctionnel. Si elle est facultative, le mot clé OPTIONAL suit le type ASN.1.

4.5.2.1.3 Erreurs associées

Les productions ASN.1 ci-après montrent comment spécifier des opérations qui notifient un échec:

```
Errors ::= "ERRORS" "{ " ErrorNames" }" | empty
ErrorNames ::= ErrorList | empty
ErrorList ::= Error | ErrorList "," Error
Error ::= value (ERROR) | type
```

Si l'opération notifie un échec, le mot clé ERRORS doit être inclus et suivi de la liste des erreurs associées, sinon le mot clé ne doit pas être présent. Les erreurs incluses dans la liste peuvent être indiquées soit à l'aide d'une référence de type ou d'une référence de valeur (c'est-à-dire un code d'erreur).

4.5.2.1.4 Spécification d'opérations corrélées

Les productions ASN.1 ci-après montrent comment spécifier des opérations corrélées:

```
LinkedOperations ::= "LINKED" "{ "LinkedOperationNames "}" | empty
LinkedOperationNames ::= OperationList | empty
OperationList ::= Operation | OperationList "," Operation
Operation ::= value (OPERATION) | type
```

Si l'opération est l'opération mère d'un ensemble d'opérations corrélées, le mot clé LINKED doit être inclus suivi de la liste des opérations filles. Les opérations filles figurant sur la liste peuvent être indiquées soit à l'aide d'une référence de type, soit d'une référence de valeur (c'est-à-dire un code d'opération).

4.5.2.2 Utilisation de la notation de valeur

La notation de valeur pour une opération est soit la notation de la valeur d'un élément de type INTEGER ou la notation d'un élément de type OBJECT IDENTIFIER. Cela dépend si l'on a attribué à l'opération une valeur locale ou une valeur globale.

4.5.2.3 Spécification de temporisateurs

La valeur du temporisateur associé à un type d'opération doit être indiquée en tant que commentaire ASN.1 par rapport à la description du type d'opération MACRO ASN.1.

4.5.3 Utilisation de la notation ERROR MACRO

La notation du type pour une erreur est le mot clé ERROR suivi à titre facultatif du mot clé PARAMETER et du type de paramètre ParameterType associé à l'information qui peut être envoyée en tant que paramètre d'erreur. Le mot clé PARAMETER ne doit pas être présent si aucune information n'est liée à la condition d'erreur.

Bien que le paramètre erreur soit toujours un élément facultatif du composant retour erreur Return Error, la spécification du type de paramètre ParameterType indique si la présence de ce paramètre est

obligatoire ou facultative d'un point de vue fonctionnel. Si elle est facultative, le mot clé OPTIONAL suit le type ASN.1.

La notation de la valeur pour une erreur est soit la notation de la valeur d'un élément du type INTEGER, soit la notation d'un élément du type OBJECT IDENTIFIER. Cela dépend si l'on a attribué à l'opération une valeur locale ou globale.

4.5.4 Utilisation de la notation CLASS (classe d'objets informationnels)

Le remplacement de la notation MACRO (définie dans la Recommandation X.209 par la notation CLASS (classe d'objets informationnels) (définie dans les Recommandations X.680 à X.683) repose sur l'idée que les applications font appel à des concepts complexes, dont les aspects doivent être exprimés sous forme de structures de données à acheminer à l'aide de protocoles durant les communications. Chacun de ces concepts est classé d'après un modèle, analogue à la définition MACRO, appelé **classe d'objets informationnels**. Ce modèle montre pour chaque classe les attributs des objets propres à cette classe. Dans les Recommandations X.680 à X.683, la notation MACRO est remplacée par la définition de la classe d'objets informationnels CLASS.

4.5.4.1 Classe (d'objets informationnels) OPERATION CLASS

La classe (d'objets informationnels) OPERATION CLASS est définie dans la Recommandation X.880. La définition ASN.1 ci-dessous présente une version légèrement modifiée de cette classe, de laquelle ont été supprimés les champs ne se rapportant pas aux applications utilisant le TC.

La notation définit le modèle d'une classe d'objets (opérations distantes) auxquels est attribué le nom OPERATION, qui comprend dix champs. Chaque champ commence par une perluète (&), qui est une indication de cette propriété, et est suivi d'un nom de champ commençant par une lettre minuscule ou majuscule. Cette distinction facilite l'identification du type de données dont peuvent être garnis les champs au moment où est définie une instance de cette classe.

Les mots entièrement en lettres majuscules font fonction de mots clés normalisés, comme CLASS et UNIQUE, ainsi que de noms de classes d'objets comme OPERATION et ERROR. Un nom de champ commençant par une lettre majuscule peut recevoir un type ASN.1 arbitraire (par exemple &Argument, voir ASN.1 ci-dessous), un ensemble d'objets informationnels (par exemple &Errors) ou un ensemble de valeurs d'un type quelconque. Dans le cas d'un ensemble d'objets informationnels, le descripteur de la classe à laquelle ces objets appartiennent suit. Un nom de champ commençant par une lettre minuscule (par exemple &returnResult) prend quant à lui la *valeur* du type ASN.1 (par exemple BOOLEAN) ou d'une classe d'objets informationnels qui suit. Un champ marqué OPTIONAL ne doit pas nécessairement être garni au moment où sont définies les instances de la classe. Le mot clé UNIQUE suivant un champ (par exemple &operationCode) signifie que ce champ est un "pointeur" qui permet d'identifier les instances de la classe en question et qui doit par conséquent rester unique dans un ensemble où une collection donnée de tels objets.

Enfin, les concepteurs de la notation ASN.1 ont autorisé dans une mesure limitée la notation définie par l'utilisateur au moyen de l'utilisation de la construction WITH SYNTAX ajoutée à la définition de la classe d'objets informationnels. Cette notation est plus commode pour l'utilisateur pour définir les instances d'une CLASS. Dans la Recommandation X.880, la syntaxe définie par l'utilisateur pour les classes OPERATION CLASS et ERROR CLASS a été délibérément choisie de manière à ressembler de très près à la notation MACRO précédente. Les (petites) modifications qu'exige la conversion d'une opération en cours ou d'une définition d'erreur de la définition MACRO à une opération en cours ou une définition d'erreur employant les définitions de la classe d'objets informationnels CLASS sont indiquées au sous-paragraphe 4.5.7.

```
OPERATION ::= CLASS
{
                                &ArgumentType              OPTIONAL,
                                &argumentTypeOptional      BOOLEAN OPTIONAL,
                                &ResultType                OPTIONAL,
                                &resultTypeOptional        BOOLEAN OPTIONAL,
                                &returnResult              BOOLEAN DEFAULT TRUE,
                                &Errors                    ERROR OPTIONAL,
                                &Linked                    OPERATION OPTIONAL,
                                &synchronous               BOOLEAN DEFAULT FALSE,
                                &alwaysReturns             BOOLEAN DEFAULT TRUE,
                                &operationCode  Code UNIQUE OPTIONAL
}
WITH SYNTAX
{
                                [ARGUMENT                  &ArgumentType [OPTIONAL
                                &argumentTypeOptional]]
                                [RESULT                    &ResultType [OPTIONAL
                                &resultTypeOptional]]
                                [RETURN RESULT             &returnResult]
                                [ERRORS                          &Errors]
                                [LINKED                    &Linked]
                                [SYNCHRONOUS                     &synchronous]
                                [ALWAYS RESPONDS           &alwaysReturns]
                                [CODE                      &operationCode]
}
Code ::= CHOICE
                                {
                                        local      INTEGER,
                                        global     OBJECT IDENTIFIER
                                }
```

4.5.4.1.1 Spécification de l'argument de l'opération

Le champ &Argument est un champ de type facultatif dans lequel on peut placer un type ASN.1 arbitraire pour définir l'argument de l'opération distante invoquée (lancée). Le concepteur de l'application indique le type à utiliser au moment où il définit une opération spécifique. Dans la syntaxe définie par l'utilisateur, le type d'argument associé suit le mot clé ARGUMENT.

Le champ &ArgumentTypeOptional, dont la valeur est facultative, prend la valeur TRUE si un argument défini peut éventuellement être absent d'un point de vue fonctionnel, ou la valeur FALSE si cet argument doit toujours être présent. Dans la syntaxe définie par l'utilisateur, ces deux cas de figure sont représentés par les mots clés OPTIONAL TRUE (ou OPTIONAL FALSE) à la suite de la définition du type du mot clé ARGUMENT.

4.5.4.1.2 Spécification du résultat de l'opération

Le champ &ResultType est un champ de type dans lequel on peut placer un type ASN.1 arbitraire pour définir le résultat de l'exécution de l'opération distante. Le concepteur de l'application indique le type à utiliser pour le résultat au moment où il définit une opération spécifique. Dans la syntaxe définie par l'utilisateur, le type du résultat associé suit le mot clé RESULT.

Le champ &resultTypeOptional, dont la valeur est facultative, prend la valeur TRUE si un résultat défini peut éventuellement être absent d'un point de vue fonctionnel, ou la valeur FALSE si ce résultat doit toujours être présent. Dans la syntaxe définie par l'utilisateur, ces deux cas de figure sont représentés par les mots clés OPTIONAL TRUE (ou OPTIONAL FALSE) à la suite de la définition du type du mot clé RESULT.

4.5.4.1.3 Spécification de résultats positifs

Le champ &returnResult, dont la valeur est facultative, est un champ du type BOOLEAN qui indique si le résultat positif d'une opération distante est toujours notifié. Dans la syntaxe définie par l'utilisateur, ce champ est identifié par le mot clé RETURN RESULT. L'absence de ce champ suppose que l'opération renvoie toujours un résultat.

NOTE – Ce champ permet de déterminer explicitement si le succès d'exécution d'une opération appelle un retour-résultat, même si aucun type de résultat n'est défini à cet effet.

4.5.4.1.4 Erreurs associées

Le champ &Errors est un champ facultatif dans lequel on peut placer des objets (erreurs) définis par la classe d'erreur CLASS ERROR (voir 4.5.5) pour définir l'ensemble d'erreurs pouvant être renvoyées en cas d'échec de l'opération distante. Dans la syntaxe définie par l'utilisateur, l'ensemble d'erreurs, s'il est présent, est mis entre accolades "{…}" et suit le mot clé ERRORS.

4.5.4.1.5 Spécification d'opérations liées

Le champ &Linked est un champ facultatif dans lequel on peut placer l'ensemble d'objets (d'opérations) définis par la classe d'opération CLASS OPERATION pour définir l'ensemble d'opérations pouvant être liées à l'opération que l'on s'emploie à définir. Dans la syntaxe définie par l'utilisateur, l'ensemble d'opérations liées, s'il est présent, est mis entre accolades "{…}" et suit le mot clé LINKED.

4.5.4.1.6 Nature synchrone de l'opération

Le champ &synchronous, dont la valeur est facultative, prend la valeur TRUE pour une opération synchrone (c'est-à-dire que l'invocateur doit attendre le résultat de l'opération avant d'en invoquer une autre pour le même exécutant) ou la valeur FALSE pour une opération non synchrone. Dans la notation définie par l'utilisateur, le code d'opération suit le mot clé SYNCRHONOUS.

NOTE – Dans le cas d'une opération synchrone, le champ &alwaysReturns (voir 4.5.4.1) doit avoir la valeur TRUE.

Les opérations d'utilisateurs de TC sont asynchrones. Ce champ a été défini de manière à prendre la valeur par défaut FALSE (c'est-à-dire asynchrone) en cas d'absence de ce mot clé; il n'y a donc pas lieu de se soucier de la présence de ce champ dans les applications des utilisateurs de TC existantes et futures. D'autres opérations, comme celles de rattachement bind et de détachement unbind (voir 5.2.2.1 et 5.2.2.2) utilisent toutefois explicitement ce champ dans leurs définitions.

4.5.4.1.7 Code d'opération

Le champ &operationCode, dont la valeur est facultative, peut être un nombre entier (une valeur unique à l'échelon local) ou un identificateur d'objet OBJECT IDENTIFIER (une valeur unique à l'échelon mondial); la valeur retenue pour ce champ doit différer des valeurs de champ des autres opérations d'un ensemble donné d'opérations. Dans la notation définie par l'utilisateur, le code d'opération suit le mot clé CODE.

S'il n'est pas défini de valeur de code d'opération, cela signifie que l'opération en question ne peut pas être invoquée à l'aide d'une unité PDU d'invocation Invoke. Un exemple d'une telle opération est l'opération de rattachement bind (voir 5.2.2.1) qui est invoquée à l'aide de l'unité PDU d'invocation de rattachement définie au 5.2.2.1.2.

4.5.4.1.8 Classes d'opération

Le champ &alwaysReturns, dont la valeur est de type BOOLEAN, indique si le résultat de l'opération est toujours renvoyé ou non. Dans la syntaxe définie par l'utilisateur, ce champ est

désigné par la valeur vrai ou faux associée au mot clé ALWAYS RETURNS. L'absence de ce champ suppose que le résultat de l'opération est toujours renvoyé.

Une opération dont le résultat est toujours renvoyé est une opération de classe 1, de classe 2 ou de classe 3. Le champ &returnResult aide à déterminer si l'opération considérée est de classe 1 ou de classe 3. La présence ou l'absence du champ &Errors restreignent respectivement le choix précédent à la classe 1 ou à la classe 3.

4.5.4.1.9 Spécification de valeurs de temporisation

On ne dispose d'aucun moyen de spécifier sous forme de notation des valeurs de temporisation pour les opérations sauf sous forme de commentaires en ASN.1.

4.5.5 Classe (d'objets informationnels) ERROR CLASS

La classe (d'objets informationnels) ERROR CLASS est définie dans la Recommandation X.880. La description ASN.1 ci-dessous présente une version légèrement modifiée de cette classe, de laquelle un champ ne se rapportant pas aux applications utilisant le TC a été supprimé.

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

4.5.5.1 Spécification du paramètre associé à une erreur

Le champ &ParameterType est un champ de type facultatif dans lequel on peut placer un type ASN.1 arbitraire pour définir le paramètre associé à une notification d'erreur pour un lancement d'opération. Le concepteur de l'application indique le type à utiliser au moment où il définit une erreur spécifique. Dans la syntaxe définie par l'utilisateur, le type du paramètre associé suit le mot clé PARAMETER.

Le champ &parameterTypeOptional, dont la valeur est facultative, prend la valeur TRUE si un paramètre défini peut éventuellement être absent d'un point de vue fonctionnel, ou la valeur FALSE si ce paramètre doit toujours être présent. Dans la syntaxe définie par l'utilisateur, ces deux cas de figure sont représentés par les mots clés OPTIONAL TRUE (ou OPTIONAL FALSE) à la suite de la définition du type du mot clé PARAMETER.

4.5.5.2 Code d'erreur

Le champ &errorCode est un champ de valeur qui peut être un nombre entier (une valeur unique à l'échelon local) ou un identificateur d'objet OBJECT IDENTIFIER (une valeur unique à l'échelon mondial); la valeur choisie pour ce champ doit être différente des valeurs des autres erreurs de tout ensemble donné d'erreurs. Dans la notation définie par l'utilisateur, le code d'erreur suit le mot clé CODE.

S'il n'est pas défini de valeur de code d'erreur, cela signifie que cette erreur ne peut pas être renvoyée à l'aide de l'unité PDU de retour d'erreur Return Error. Un exemple d'une telle erreur est l'erreur indiquant la capacité de rattachement bind (voir 5.2.2.1) qui est renvoyée à l'aide de l'unité PDU d'erreur de rattachement définie au 5.2.2.1.2.

4.5.6 Exemples de descriptions d'opérations et d'erreurs

Le présent sous-paragraphe explique une spécification de protocole qui traite des définitions des opérations et des erreurs associées pour un simple ASE utilisateur du TC. L'objet des opérations et des erreurs est brièvement décrit sous forme textuelle. Ensuite, les opérations et les erreurs, ainsi que les types de données connexes, sont officiellement décrits dans un module ASN.1.

L'exemple ci-après est fondé sur un dialogue du type libre appel fictif, entre un centre de commutation et une base de données libre appel.

4.5.6.1 Objet des opérations et des erreurs

4.5.6.1.1 Fournir des informations d'acheminement

Cette opération est lancée par un centre de commutation pour demander à une entité éloignée de fournir des informations d'acheminement afin d'établir une communication pour un abonné. Les informations d'acheminement fournies peuvent être un numéro de destination et peuvent dépendre du numéro du demandeur et/ou du service de base demandé. Dans le dernier cas, l'opération fille getCallingPartyNumber (obtenir le numéro du demandeur) est lancée.

4.5.6.1.2 Obtenir le numéro du demandeur

Cette opération est lancée par un élément de réseau afin de demander à un centre de commutation de fournir le numéro du demandeur lié à l'établissement d'une communication.

4.5.6.1.3 Numéro appelé non valable

Cette erreur est renvoyée par un élément du réseau pour indiquer que le numéro appelé reçu n'est pas conforme au plan de numérotage utilisé.

4.5.6.1.4 Abonné non atteignable

Cette erreur est renvoyée par un élément de réseau pour indiquer qu'il n'y a actuellement aucune information d'acheminement disponible correspondant au numéro appelé.

4.5.6.1.5 Appel interdit

Cette erreur est renvoyée par un élément du réseau pour indiquer que la communication ne peut pas être établie car le numéro du demandeur n'est pas compatible avec les conditions d'interdiction liées au numéro appelé.

4.5.6.1.6 Numéro du demandeur non disponible

Cette erreur est renvoyée par un centre de commutation pour indiquer que le numéro du demandeur ne peut pas être fourni.

4.5.6.1.7 Erreur de traitement

Cette erreur est renvoyée par un élément du réseau pour signaler une erreur de traitement.

4.5.6.2 Spécification ASN.1

Le module ASN.1 ci-après spécifie les opérations et les erreurs connexes ainsi que les types de données qui correspondent aux éléments de protocole décrits ci-dessus. Dans cet exemple, les opérations et les erreurs sont définies par une valeur utilisant une production "assignation de valeur" ASN.1.

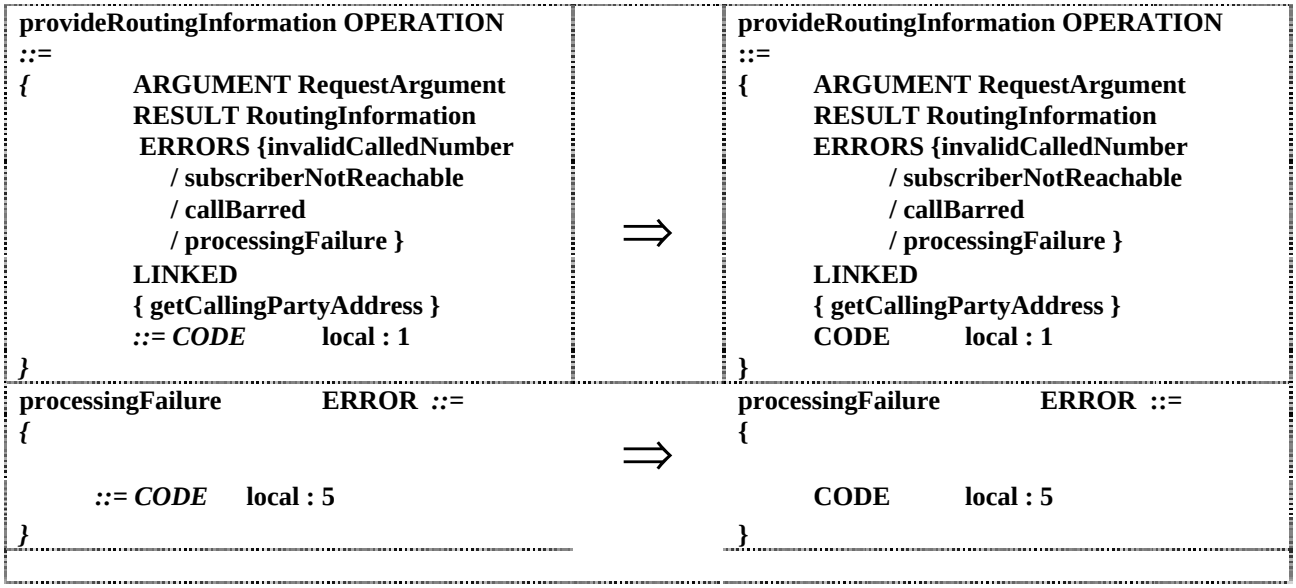
```
TCAP-Examples { ccitt recommendation q 775 modules(2) examples(2) version1(1) }
DEFINITIONS ::=
BEGIN

IMPORTS OPERATION, ERROR
FROM TCAPMessages { ccitt recommendation q 773 modules(2) messages(1) version2(2) };

provideRoutingInformation                OPERATION
ARGUMENT                                 RequestArgument

RESULT                                   RoutingInformation

ERRORS                                   { invalidCalledNumber,
                                         subscriberNotReachable,
                                         callBarred,
                                         processingFailure }

LINKED                                   { getCallingPartyAddress }
-- temporisateur T-pi = 10 s
::= localValue : 1

getCallingPartyAddress                   OPERATION
RESULT                                   CallingPartyAddress

ERRORS                                   { callingPartyAddressNotAavailable,
                                         processingFailure }

-- temporisateur T-gp = 5 s
::= localValue : 2

invalidCalledNumber ERROR ::= localValue : 1
subscriberNotReachable ERROR ::= localValue : 2
calledBarred ERROR ::= localValue : 3
callingPartyAddressNotAvailable ERROR ::= localValue : 4
processingFailure ERROR ::= localValue : 5
-- types de données

RequestArgument  ::= SEQUENCE {
calledNumber            IsdnNumber,
basicService        BasicServiceIndicator OPTIONAL
}

RoutingInformation ::= CHOICE {
        reroutingNumber         [0] IMPLICIT IsdnNumber,
        forwardedToNumber       [1] IMPLICIT IsdnNumber }

BasicServiceIndicator  ::= ENUMERATED {
speech (0),
unrestrictedDigital (1) }

CallingPartyAddress ::= IsdnNumber

IsdnNumber  ::= SEQUENCE {
typeOfAddress           TypeOfAddress,
digits                  TelephonyString }
```

```
TypeOfAddress ::= ENUMERATED {
national (0),
international (1),
private (2) }

TelephonyString ::= IA5String  (FROM ("0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"|"*"|"#")) (SIZE (1..15))

END
```

4.5.7 Passage de la notation MACRO à la notation CLASS (classe d'objets informationnels)

Toutes les spécifications d'utilisateurs de TC présentées jusqu'ici utilisent la notation MACRO ASN.1 définie dans la Recommandation X.208. Le présent sous-paragraphe indique les modifications à opérer pour passer de l'utilisation de ces macro-instuctions à l'utilisation de la notation de la classe d'objets informationnels CLASS. Comme indiqué au 4.5.4, la syntaxe définie par l'utilisateur des classes OPERATION CLASS/ERROR CLASS a été expressément choisie dans la Recommandation X.880 pour présenter la plus grande similitude possible avec la construction précédente. Du point de vue d'un utilisateur, de petites modifications s'imposent, parmi lesquelles un repositionnement de différents symboles et l'apport de plus amples précisions dans la notation des aspects d'une opération qui, dans la notation MACRO, ne pouvaient être exprimés que sous forme de commentaires.

Dans le présent sous-paragraphe, l'opération *provideRoutingInformation* et l'erreur *processingFailure* spécifiées au 4.5.6.2 sont représentées dans les deux notations. Dans chaque cas, il convient de supprimer de la partie gauche, qui représente la notation en macro-instructions, les symboles soulignés pour obtenir la nouvelle notation (partie droite), et d'insérer les symboles en ***italique gras*** dans l'ancienne notation pour obtenir la nouvelle notation.



Utilisation de modules ASN.1

Un module est une construction ASN.1 dans laquelle un concepteur de protocole réunit plusieurs définitions de types et de valeurs.

Théoriquement, l'ASN.1 n'impose aucune limitation du nombre de modules utilisés pour définir un protocole. Toutes les définitions peuvent être contenues dans un ou plusieurs module(s). Toutefois, si les définitions contenues dans un module sont nécessaires dans un autre (par exemple, l'erreur utilisée pour une opération est définie dans un autre module), la définition correspondante devient alors disponible en l'EXPORTant du module dans lequel elle est définie et en l'IMPORTant dans le

module dans lequel elle est utilisée. Cela s'applique à tous les objets ASN.1, qu'ils soient définis selon le type ou la valeur.

Cette possibilité donne au concepteur d'application toute latitude de structurer les modules conformément aux besoins ou à des règles propres. Par exemple, un seul modèle peut contenir toutes les définitions, en particulier dans un environnement AE, ou ASE uniques. En revanche, il peut y avoir un seul module pour chaque définition d'un élément ASE, chaque module contenant toutes les opérations et les erreurs utilisées exclusivement par cet élément. A l'autre extrémité, toutes les opérations et erreurs peuvent être définies dans un module du registre central et exportées pour être utilisées dans les autres modules dans lesquels les éléments ASE sont définis.

Il peut être nécessaire d'utiliser un "mode mixte" de notation ASN.1 pour la définition de modules. Cette nécessité découle de l'utilisation de la notation ASN.1:1994 définie dans les Recommandations X.680 à X.683 pour certains cas où les définitions sont reprises de Recommandations UIT-T existantes employant la nouvelle notation. Un exemple de tels cas est l'utilisation de certains concepts du service d'opérations distantes ROSE (Recommandation X.880) comme les définitions statiques de contextes d'application (voir 5.1.1), et l'utilisation des opérations d'annuaire X.500 pour certaines interfaces du réseau intelligent. Il importe de veiller à ce que les anciens modules écrits en ASN.1:1989 n'aient pas à être réécrits et qu'ils puissent coexister avec les modules écrits selon les constructions de la notation ASN.1:1994.

Des directives détaillées concernant le fonctionnement dans ce "mode mixte" sont données dans la Recommandation X.680; elles sont résumées ici:

- une spécification peut comporter des modules en ASN.1:1989 et en ASN.1:1994. Toutefois, un module donné DOIT être conforme soit à la notation de 1989 soit à celle de 1994; des commentaires peuvent être inclus pour indiquer la version de la notation utilisée dans chacun des modules;
- des références de type et de valeur d'un module en ASN.1:1989 peuvent être importées dans un module en ASN.1:1994 à condition:
  a) que des définitions MACRO en ASN.1:1989 ne soient PAS importées dans un module utilisant la notation ASN.1:1994;
  b) que des identificateurs pour les valeurs SET, SEQUENCE et CHOICE soient présents;
- des références de type et de valeur d'un module en ASN.1:1994 peuvent être importées dans un module ASN.1:1989 à condition:
  a) que les nouveaux types ASN.1:1994 (CHARACTER STRING**,** UniversalString, BMPString, EMBEDDED-PDV) ne soient PAS importés.

4.5.8 Attribution et gestion des codes d'opération et d'erreur

4.5.8.1 Généralités

Les sous-paragraphes 4.1 à 4.5.4 décrivent comment les contextes d'application (AC), les éléments de service d'application (ASE), les opérations et les erreurs peuvent être spécifiés. On examine également comment les éléments ASE utilisent les opérations et les erreurs, et comment les ASE proprement dits sont utilisés pour définir les contextes d'application entre deux entités d'application homologues.

Les AC et les ASE sont des outils de modélisation et de spécification commodes qui servent à élaborer des protocoles d'application. A la fin, pendant un dialogue entre deux utilisateurs de TC, tout protocole d'application utilisant un TC est constitué par l'échange de valeurs de données pour les types d'opérations et d'erreurs identifiés respectivement par leur code d'opération ou d'erreur. La seule condition imposée dans le système de signalisation n° 7 (et le service ROSE) est la suivante: les codes d'opération et d'erreur doivent être uniques dans une syntaxe abstraite. Comme il n'existe

pas actuellement de moyen de signaler explicitement la syntaxe abstraite à laquelle un code d'opération ou d'erreur appartient, le concepteur des applications doit veiller à ce que ces codes soient uniques dans le cadre d'un numéro de sous-système ou d'un contexte d'application. Dans le cas où le domaine des codes d'opération et d'erreur est un contexte d'application, il faut que le nom du contexte d'application soit acheminé à l'extrémité distante à l'aide du protocole de la portion de dialogue.

Les mécanismes d'attribution et de gestion des codes d'opération et d'erreur possibles sont nombreux, de même que les facteurs dont il faut tenir compte. La structure de l'AC/ASE et la réutilisation des opérations et des erreurs constituent deux éléments importants.

4.5.8.2 Importation et exportation d'opérations et d'erreurs

Comme n'importe quel autre type ASN.1, les opérations et les erreurs peuvent être exportées et importées entre modules ASN.1. Il est possible d'utiliser cette méthode lorsqu'il est nécessaire de définir une opération dont le type correspond à une opération existante, lorsque la valeur à attribuer à cette nouvelle opération est différente de celle qui est attribuée à l'opération existante (c'est-à-dire à des fins d'unicité). Cela est illustré par l'exemple ci-après, où l'identificateur d'objet 1 et l'identificateur d'objet 2 sont des identificateurs fictifs.

```
ExportingModule { objectIdentifier1 }  DEFINITIONS ::=
BEGIN
EXPORTS operation1, OperationTypeA, error1, ErrorTypeA;

IMPORTS OPERATION, ERROR FROM TCAPMessages
{ ccitt recommendation q 773 modules(2) messages(1) version2(2) };

operation1                  OPERATION
ARGUMENT                    ParameterType1
RESULT                      ResultType1
ERRORS                      { error1 }
::= localValue 1

OperationTypeA ::=          OPERATION
ARGUMENT                    ParameterTypeA
RESULT                      ResultTypeA
ERRORS                      { ErrorTypeA}

operation2 OperationTypeA ::= localValue : 2

error1  ERROR
PARAMETER DiagnosticType1
::= localValue : 1

ErrorTypeA ::= ERROR
PARAMETER DiagnosticTypeA

error2 ErrorTypeA ::= localValue : 2
-- Noter que ParameterType1, ResultType1, ParameterTypeA, ResultTypeA,
-- DiagnosticType1 et DiagnosticTypeA doivent être définis quelque part.
-- S'ils ne sont pas définis dans ce module, ils doivent être importés du module dans lequel ils sont définis.
END


ImportingModule { objectIdentifier2 }  DEFINITIONS ::=
BEGIN
IMPORTS OPERATION, ERROR FROM TCAPMessages
{ ccitt recommendation q 773 modules(2) messages(1) version2(2) };
operation1, OperationTypeA, error1, ErrorTypeA
FROM ExportingModule { objectIdentifier1 };
```

operation2 OPERATION
ARGUMENT ParameterTypeX -- *doit être défini quelque part dans le module*
::= localValue : 2

error2 ERROR ::= localValue : 2
-- Valeur 2 est déjà utilisée. Donc, valeur 3 est allouée aux objets importés.
operationA OperationTypeA ::= localValue : 3
errorA ErrorTypeA ::= localValue : 3

END

4.5.8.3 Incidence de la structure des ASE/AC sur la gestion des codes d'opération et d'erreur

En ce qui concerne la structure des AC/ASE, les possibilités sont les suivantes:

4.5.8.3.1 Approche monolithique – Un contexte d'application, un élément de service d'application

Au niveau du concept, c'est l'approche la plus simple. Le protocole d'application est défini par un contexte d'application qui comprend seulement un élément ASE (en plus du TC). Toutes les opérations utilisées dans cet élément pourront être définies dans un module ASN.1, qui contient également la définition du lot d'opérations de l'ASE et de l'AC. A l'intérieur du protocole, toutes les opérations et les erreurs sont identifiées de façon univoque en recevant une valeur locale unique (nombre entier).

L'avantage de ce schéma est sa simplicité. Son inconvénient est qu'il ne permet pas d'identifier des blocs fonctionnels indépendants qui peuvent évoluer séparément dans le cadre de la définition de l'AC.

4.5.8.3.2 Un contexte d'application comportant plusieurs éléments de service d'application (ASE)

Lorsqu'il définit un protocole d'application, le concepteur doit choisir de structurer le contexte d'application de façon qu'il comporte deux ou plusieurs éléments ASE. Par exemple, on pourra décider de regrouper les éléments de protocole liés à l'authentification de l'usager dans un élément ASE distinct (qui pourrait être réutilisé dans un autre protocole), et ceux qui sont associés à l'interrogation de la base de données dans un autre élément ASE. Cela peut faciliter la conception d'ensemble des systèmes, mais, lorsque tous les ASE constituants sont regroupés pour former le contexte d'application, il faudra s'assurer que l'on n'a pas attribué la même valeur aux différentes opérations/erreurs contenues dans les différents éléments ASE.

Utiliser la même opération/erreur dans deux éléments ASE différents à l'intérieur du même contexte d'application ne pose pas de problème. Si les valeurs attribuées à cette même opération sont les mêmes dans chaque cas, il y aura dans le protocole une seule opération/erreur associée à cette valeur. Si différentes valeurs sont attribuées, il apparaîtra dans le protocole qu'il y a deux opérations/erreurs différentes mais au niveau de la mise en œuvre de l'application, ces deux valeurs différentes provoqueront le lancement/l'identification de la même opération/erreur.

Toutefois, si à l'intérieur de ce même contexte d'application, une opération définie dans un élément ASE reçoit la même valeur qu'une opération différente dans un autre élément ASE, cela posera de toute évidence un problème. Lorsqu'un élément ASE est utilisé dans un seul contexte d'application, un schéma d'attribution de code simple permet de surmonter ce problème. Mais, lorsque le même élément ASE est utilisé dans plusieurs contextes d'application, cette situation peut devenir difficile à gérer, et les seules approches sûres sont celles qui sont décrites sous i) à iii) ci-après.

i) deux ou plusieurs protocoles partagent les mêmes valeurs d'opération/erreur locales:

lorsque les éléments ASE sont définis, les valeurs sont attribuées par le concepteur de protocole de manière à éviter toute incompatibilité de valeurs. Une coordination s'impose pour définir les ASE. En d'autres termes, les ASE partagent la même syntaxe abstraite;

ce système présente un inconvénient dans le cas où l'un des ASE est utilisé dans plusieurs contextes (c'est-à-dire avec un ensemble différent d'ASE), car il est alors plus ou moins impossible d'éviter des problèmes de compatibilité de valeurs pour une combinaison.

ii) assignation de valeurs globales (identificateurs d'objet) pour des opérations et des erreurs:

étant donné qu'il y a un seul identificateur d'objet dans le système de signalisation n° 7, il n'y a aucun risque d'incompatibilité de valeurs quand un élément ASE est combiné à un autre;

l'inconvénient de ce schéma est que l'identificateur d'objet, s'il est codé, est plus long qu'un nombre entier simple;

iii) partage des opérations/erreurs par l'assignation de types et non de valeurs lors de la définition des opérations/erreurs:

on suppose ici que le type des opérations et des erreurs a été défini indépendamment de la valeur assignée;

quand un concepteur de protocole définit un contexte d'application, il rassemble tous les types d'opérations et d'erreurs utilisés par les ASE requis et attribue des valeurs adaptées de manière à éviter toute incompatibilité;

on peut considérer que ce faisant, le concepteur de protocole définit un nouvel ensemble d'ASE isomorphes qui diffèrent simplement des ASE existants par les valeurs de leurs opérations et erreurs.

4.5.8.4 Réutilisation des opérations et des erreurs

Indépendamment du nombre d'ASE inclus dans un protocole, il est parfois utile d'inclure une opération ou une erreur au moment de définir un nouvel ASE.

L'opération ou l'erreur peut être réutilisée de la manière suivante.

L'opération est importée dans l'un des modules définissant l'un des ASE. Cette importation n'est pas possible lorsque les valeurs sont différentes.

Ces blocs peuvent être identifiés si:

i) il existe un registre central des opérations et des erreurs dont les valeurs sont prises dans une fourchette réservée qui n'est jamais utilisée par des opérations spécifiques d'ASE. Cette solution impose une contrainte aux ASE utilisateurs de TC qui risque de subsister dans un environnement plus vaste (c'est-à-dire si les opérations ou les erreurs des protocoles DSS 1 ou ISO doivent être utilisées);

ii) des valeurs globales ont été attribuées aux opérations et aux erreurs. Cette approche présente un inconvénient dans la mesure où une valeur globale nécessite davantage d'octets pour être codée qu'une valeur locale et doit être officiellement enregistrée dans un arbre d'identificateur d'objet;

iii) le type d'opération ou d'erreur est importé dans l'un des modules définissant l'un des éléments ASE dans lesquels une valeur adaptée est attribuée. Cela suppose que le protocole d'exportation utilise la méthode à deux étapes pour définir les opérations et les erreurs ou que les types d'opération et d'erreur requis figurent dans un registre central;

iv) l'opération ou l'erreur est entièrement redéfinie bien qu'une partie de la définition d'origine (par exemple le type de l'argument) puisse être importée.

4.6 Spécifications de types de données

4.6.1 Généralités

Comme indiqué au paragraphe ci-dessus, le type d'information qui peut accompagner un lancement d'opération, à savoir la notification d'une réussite ou d'un échec, est spécifié en tant que type de données ASN.1. Cela vaut aussi pour l'information qui peut être échangée sous forme de données d'utilisateur de la portion de dialogue.

Ce type de données peut être du type intégré (par exemple, integer, boolean, null, octet, string, etc.) ou du type structuré (par exemple, sequence, sequence-of type, choice type, etc.). Il peut aussi être inspiré de ces types par un mécanisme de construction de sous-type (par exemple, contrainte de taille, gamme de valeurs) ou d'étiquetage.

4.6.2 Utilisation d'étiquettes

La notation ASN.1 fournit un mécanisme d'étiquetage qui permet de définir un type isomorphe à un type existant et qui ne diffère que par la présence de son étiquette.

Comme indiqué clairement dans la Recommandation X.208 (ASN.1), les étiquettes sont destinées à une utilisation en machine pour faciliter avant tout le processus de décodage.

Les étiquettes ne doivent pas être utilisées pour l'identification directe des éléments d'information, étant donné qu'elles sont vues du point de vue d'un processus d'application local. La manière dont ces éléments d'information sont identifiés localement est une question de mise en œuvre qui dépend de la conception du logiciel et du langage utilisé pour manipuler la représentation des données internes. A cet égard, il convient de noter que des étiquettes distinctes sont surtout nécessaires dans l'une des situations suivantes:

– les éléments d'information font partie d'un ensemble (non ordonné) (c'est-à-dire un type d'ensemble) et par conséquent leur position relative ne peut être utilisée pour différencier deux éléments d'information du même type (ayant donc la même étiquette);
– les éléments d'information font partie d'un ensemble ordonné (c'est-à-dire un type de séquence) mais la présence ou l'absence d'éléments facultatifs ne permet pas de faire une distinction entre la présence d'un élément facultatif et la présence d'un élément d'information du même type suivant immédiatement;
– le même type de base apparaît deux fois dans un type de choix.

Il y a quatre classes d'étiquettes. Outre la classe universal qui est utilisée pour identifier un type de base, trois classes sont définies et permettent la définition de types isomorphes à des fins de décodage:

– la *classe APPLICATION-WIDE* – Les étiquettes attribuées à cette classe peuvent servir à identifier directement la structure du type de données à décoder. Elles sont significatives dans une application et ne doivent pas être utilisées s'il y a risque d'incompatibilité entre les valeurs. La classe APPLICATION-WIDE ne devrait être utilisée que si l'application est un domaine "fermé" ou s'il existe un registre commun;
– la *classe CONTEXT-SPECIFIC* – Les étiquettes attribuées à cette classe ne sont significatives que dans un domaine défini. C'est pourquoi, le processus de décodage identifie la structure des données à décoder à la fois à partir de la valeur de l'étiquette et du contexte dans laquelle elle apparaît. On estime habituellement que le contexte est limité à la prochaine construction supérieure;
– la *classe PRIVATE* – Analogue à la classe APPLICATION-WIDE mais n'entre pas dans le cadre de la normalisation.

Il convient de noter que la classe CONTEXT-SPECIFIC est la seule (si elle est utilisée correctement) qui garantit qu'il n'y aura jamais d'incompatibilité entre les valeurs, lorsqu'il y a des importations/exportations de types de données entre modules ou protocoles.

4.6.3 Instances et types

Il est nécessaire de différencier clairement un type de données d'une instance d'un type de données (c'est-à-dire les véritables éléments d'information acheminés dans un message ou une sous-structure). A des fins de spécification et de facilité de lecture, l'ASN.1 fournit une notation type nommé NamedType qui permet de qualifier une instance spécifique d'un type de données à l'aide d'un identificateur ASN.1.

Il convient de noter qu'il n'est pas nécessaire de définir un type de données par élément d'information. Lorsque deux éléments d'information sont équivalents du point de vue de la syntaxe, il est de toute évidence plus pratique de les représenter en tant que deux instances du même type de données, ou si cela est nécessaire aux fins de décodage, en tant qu'instances de deux types dérivés du même type de données grâce à l'étiquetage CONTEXT-SPECIFIC, et dont la définition apparaîtra donc seulement dans la définition de la construction supérieure (par exemple, les éléments étiquetés ne sont définis que dans le contexte spécifique de la construction supérieure).

4.6.4 Exportation et importation de types de données

Il se pourrait que des protocoles de signalisation utilisant le TC aient à utiliser des éléments d'information définis dans d'autres spécifications de protocole de signalisation. Plutôt que de définir un nouvel élément d'information, il est préférable d'importer le type associé des spécifications dans lesquelles il a été défini pour la première fois. Des types de données peuvent être importés officieusement ou officiellement selon la façon dont le protocole d'exportation est spécifié.

– le protocole d'exportation est spécifié à l'aide d'un module ASN.1, qui exporte les types de données requis: les types de données requis peuvent être importés officiellement dans l'un des modules qui définit le nouveau protocole.

– le protocole d'exportation n'est pas spécifié à l'aide d'un module ASN.1: un moyen pratique est de définir un type de données isomorphe au type de chaîne d'octet et d'en spécifier officieusement la structure interne à l'aide d'une référence à la spécification dans laquelle il est défini (c'est-à-dire en utilisant un énoncé de commentaire).

4.7 Comment spécifier les syntaxes abstraites

Les spécifications relatives à l'ASE et à l'AC impliquent qu'il soit fait référence à une ou plusieurs syntaxes abstraites. Chaque syntaxe représente, au niveau de l'abstraction (c'est-à-dire indépendamment des techniques de codage) des ensembles de valeurs de données qui peuvent être échangées pendant la communication.

Il est actuellement inutile de donner explicitement un nom à la syntaxe abstraite formée par les messages de TC pour une application donnée car cette syntaxe abstraite est implicitement identifiée par le numéro de sous-système qui traite l'AE. Toutefois, la structure de l'information d'utilisateur acheminée dans la portion de dialogue doit être définie dans le cadre d'une ou de plusieurs autres syntaxes abstraites.

En conséquence, un concepteur de protocole qui souhaite avoir des informations d'utilisateur qui ne constituent pas des composants devant être acheminés par le TC doit commencer par définir une ou plusieurs syntaxes abstraites comprenant tous les types de données dont les valeurs peuvent être acheminées.

Il doit aussi donner un nom à chacune de ces syntaxes abstraites. Ce nom qui est une valeur du type OBJECT IDENTIFIER (IDENTIFICATEUR D'OBJET) servira de référence directe lorsque la valeur réelle sera acheminée dans le cadre d'une valeur construite du type EXTERNAL, comme il est spécifié dans la Recommandation Q.773.

Il n'existe pas actuellement de méthode officielle pour spécifier une syntaxe abstraite; toutefois lorsqu'une syntaxe de ce type peut être décrite au moyen de la notation ASN.1, la méthode la plus simple consiste à définir un type de choix élaboré à partir de tous les types de données qui forment la syntaxe abstraite.

Une syntaxe abstraite peut alors être définie officieusement par la phrase ci-après qui doit figurer dans les spécifications de protocole:

"l'ensemble des valeurs de données du type module X. Le type A forme une syntaxe abstraite qui est identifiée par le nom de syntaxe abstraite suivant: <objectIdentifierValue>".

Dans la phrase précédente, le type A correspond au type de choix et le module X au module dans lequel il est défini.

Dans ce contexte, le nom de la syntaxe abstraite se réfère aussi implicitement aux règles de codage qu'il convient d'appliquer à la syntaxe abstraite. Ces règles de codage qui peuvent correspondre (mais pas nécessairement) à celles qui sont définies dans la Recommandation X.209, doivent être acceptées *a priori* par les utilisateurs de TC.

La classe d'objets informationnnels ABSTRACT-SYNTAX ASN.1 spécifiée dans la Recommandation X.681 peut aussi être utilisée pour la définition de syntaxes abstraites. Un exemple d'une telle utilisation est donné ci-dessous à travers la définition d'une syntaxe abstraite englobant les valeurs du type InitData qui est un ensemble de trois unités de données de protocole utilisé lors de l'établissement du dialogue pour transférer soit la liste des unités fonctionnels prises en charge, soit l'information d'authentification:

```
InitModule DEFINITIONS ::=
BEGIN

InitData ::= CHOICE {
functionalUnits    [0] IMPLICIT FunctionalUnits,
authenticationInfo  [1] IMPLICIT AuthenticationInfo }

FunctionalUnits ::= SEQUENCE OF FunctionalUnit

FunctionalUnit ::= ENUMERATED {unit(1), unit2(2), unit3(3) }

AuthenticationInfo ::= SEQUENCE {
        algorithm OBJECT IDENTIFIER,
        signature OCTET STRING }
init-abstract-syntax ABSTRACT-SYNTAX ::=
{
InitData                IDENTIFIED BY           { -- une valeur quelconque d'identificateur d'objet -- }
}

END
```

4.8 Règles de codage

La syntaxe concrète des messages du TC (c'est-à-dire le train de bits échangés entre TC homologues en tant que données d'usager des messages SCCP) est établie en appliquant les règles de codage de base à la description de la syntaxe abstraite des messages du TC [y compris les éléments d'usager du TC à l'exception de ceux qui sont acheminés en tant que valeurs du type EXTERNAL (par exemple le champ d'information d'utilisateur d'une APDU de gestion de dialogue)]. Les règles de codage de base sont décrites dans la Recommandation X.209, certaines restrictions mineures sont signalées dans la Recommandation Q.773 en ce qui concerne le codage de la partie TC.

L'information d'utilisateur acheminée en tant que valeur du type EXTERNAL peut aussi (mais pas nécessairement) être codée conformément aux règles de codage de base. Dans ce cas, le nom de la syntaxe abstraite associé sert aussi de référence implicite aux règles de codage appliquées (voir 3.3.3).

Il faut noter que les règles de codage de base permettent plusieurs options, spécialement pour le codage des longueurs. Cela veut dire qu'une mise en œuvre doit être à même de décoder une unité de données indépendamment des options de codage choisies par l'entité expéditrice.

5 Mappage des concepts génériques du service ROS et des services du TC

5.1 Aperçu général

La Recommandation X.880 définit un modèle générique de communication interactive entre objets, dans lequel l'interaction de base implique l'invocation (lancement) d'une opération par un objet (l'invocateur) et son exécution par un autre (l'exécutant). Ce modèle, appelé modèle d'opérations distantes ou modèle ROS (ROS, *remote operations*) comporte un ensemble de classes d'objets informationnels ASN.1 à l'usage des concepteurs de protocoles pour la spécification d'applications de type ROS.

La Recommandation X.800 atteste l'existence de multiples réalisations possibles de ce modèle, à des fins de communication. Le présent paragraphe a pour but de montrer comment et pourquoi le TC peut être considéré comme l'une de ces réalisations, par le mappage des concepts génériques et des services du TC.

5.1.1 Notation et concept applicables au modèle générique ROS

La Recommandation X.880 définit plusieurs classes d'objets informationnels qui sont utiles pour la spécification de protocoles d'application de type ROS. Ces classes d'objets sont définies à l'aide de la notation ASN.1 de spécification d'objets informationnels définie dans la Recommandation X.681.

La classe OPERATION est utilisée pour définir une opération. Elle équivaut simplement à la classe OPERATION MACRO définie dans les Recommandations X.219 et Q.773. Cette classe peut être utilisée par les concepteurs d'applications d'utilisateurs de TC à la place de la notation MACRO décrite au paragraphe 4. Des directives permettant de passer de la notation MACRO à la présente notation sont données dans l'Annexe C/X.880.

La classe ERROR est utilisée pour définir une opération. Elle équivaut simplement à la classe ERROR MACRO définie dans les Recommandations X.219 et Q.773. Cette classe peut être utilisée par les concepteurs d'applications d'utilisateurs de TC à la place de la notation MACRO décrite au paragraphe 4. Des directives permettant de passer de la notation MACRO à la présente notation sont données dans l'Annexe C/X.880.

La classe OPERATION-PACKAGE est utilisée pour définir un ensemble d'opérations ne pouvant être invoquées que par un objet ROS tenant le rôle "client", les opérations ne pouvant être invoquées

que par un objet ROS tenant le rôle de "serveur" et les opérations pouvant être invoquées par ces deux objets ROS. En cas d'utilisation de services de communication du système de signalisation n° 7 ou de type OSI, un lot d'opérations est réalisé en tant qu'élément de service d'application (ASE).

La classe CONNECTION-PACKAGE est utilisée pour définir les opérations de rattachement et de détachement utilisées pendant les phases d'établissement et de libération d'une association. Lorsqu'elle utilise les services de communication du système de signalisation n° 7, la réalisation d'un lot de connexion fait appel aux procédures utilisant les services de gestion de dialogue structuré du gestionnaire de transactions. Les contextes d'application qui ne nécessitent pas l'invocation explicite des opérations de rattachement et de détachement continueront d'être considérés comme incluant un lot de connexion utilisant les opérations prédéfinies de rattachement vide emptyBind et de détachement vide emptyUnbind.

La classe CONTRACT est utilisée pour définir un contrat d'association en fonction d'un lot de connexion et d'un ou de plusieurs lots d'opérations. Au moment de la spécification du contrat, il est procédé à l'identification des différents lots, à savoir ceux dans lesquels seul l'initiateur de l'association tient le rôle de client, ceux dans lesquels seul le répondeur de l'association tient le rôle de client et ceux dans lesquels aussi bien l'un que l'autre peut tenir le rôle de client. En cas d'utilisation de services de communication du système de signalisation n° 7 ou de type OSI, un contrat est réalisé en tant que contexte d'application.

La classe ROS-OBJECT-CLASS est utilisée pour définir un ensemble de capacités communes d'un ensemble d'objets ROS en fonction des contrats (d'association) acceptés par les initiateurs et les répondeurs. Dans le cadre d'une réalisation utilisant le TC ou l'OSI, un objet ROS correspond à une partie d'un processus d'application.

Ces classes offrent une notation utilisable pour la conception d'applications de type ROS indépendamment de toute réalisation particulière. La spécification du protocole effectif nécessite la définition d'un contexte d'application indiquant la manière dont le contrat d'opérations est réalisé. Le sous-paragraphe 5.4 définit une classe d'objets informationnels APPLICATION-CONTEXT utilisable pour la spécification de formes de réalisation d'un contrat d'opérations qui utilisent le TC.

5.1.2 Modèle de communication

La réalisation du service ROS implique la sélection d'un intermédiaire approprié permettant de véhiculer les invocations et les réponses entre deux objets ROS.

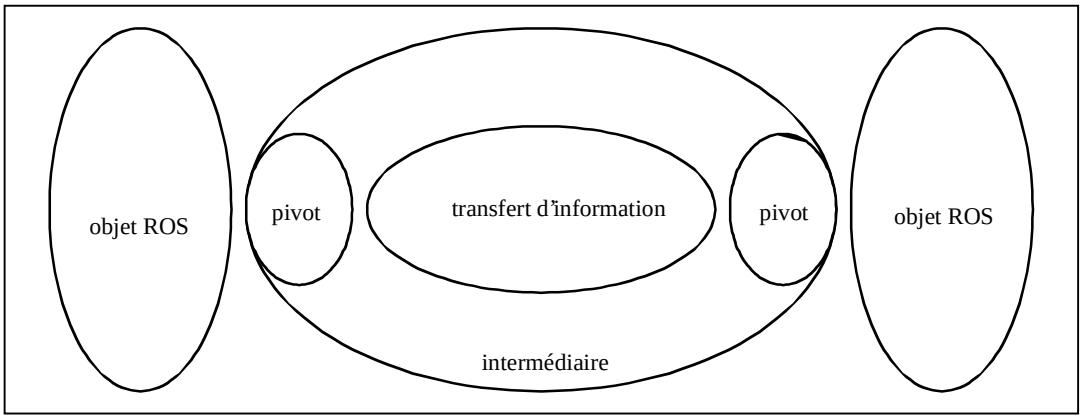
On distingue deux grandes catégories d'intermédiaires possibles:

- les intermédiaires nécessaires lorsque l'invocateur et l'exécutant doivent prendre place dans un même équipement;
- les intermédiaires nécessaires lorsque l'invocateur et l'exécutant doivent prendre place dans des équipements distincts.

Les intermédiaires de la première catégorie peuvent en outre offrir des fonctions de passation de messages et d'appel de procédures.

Les intermédiaires de la seconde catégorie dépendent du type de réseau assurant l'interconnexion des deux objets et d'un certain nombre de critères de qualité de service.

Selon le modèle de communication présenté dans la Recommandation X.880, l'intermédiaire est constitué de deux objets pivots (*stub*) (un pour l'invocateur et un pour l'exécutant) et d'un objet de transfert d'information (voir la Figure 1). Les capacités de l'objet de transfert d'information incluent en outre les fonctionnalités de commande d'association éventuellement nécessaires pour établir une association entre les entités d'application intervenant dans la communication.



T1180040-96

Figure 1/Q.775 – Modèle de communication générique ROS

Les deux objets pivots ont simplement pour rôle de transformer les invocations et les réponses qu'ils échangent entre eux à l'aide de l'objet de transfert d'information en unités de données de protocole (ou vice versa). Pour un type donné d'objets pivots, il existe plusieurs types possibles d'objets de transfert d'information.

Dans le contexte OSI, les objets pivots sont réalisés par l'élément de service d'opérations distantes (ROSE, *remote operation service element*), alors que plusieurs réalisations de transfert d'information, combinant de manière appropriée l'élément de service de contrôle d'association (ACSE), l'élément de service de transfert fiable (RTSE) et le service de présentation, sont utilisables.

Les objets pivots sont réalisés par le bloc de gestion des composants de la sous-couche composant du TC (voir la Recommandation Q.774) ainsi que par un ensemble d'éléments ASE spécifiques à l'opération (les ASE des utilisateurs de TC). La gestion des composants, dont les services sont définis au 3.1.3/Q.771, commande le protocole générique requis pour l'invocation d'opérations arbitraires et les renvois de notification correspondants.

Chaque élément ASE d'utilisateur de TC intègre la connaissance des définitions des opérations spécifiques qu'implique un lot d'opérations donné. La sous-couche composants et les éléments ASE d'utilisateurs de TC ont collectivement connaissance de toutes les opérations du contrat d'association. Voir la Figure 2.

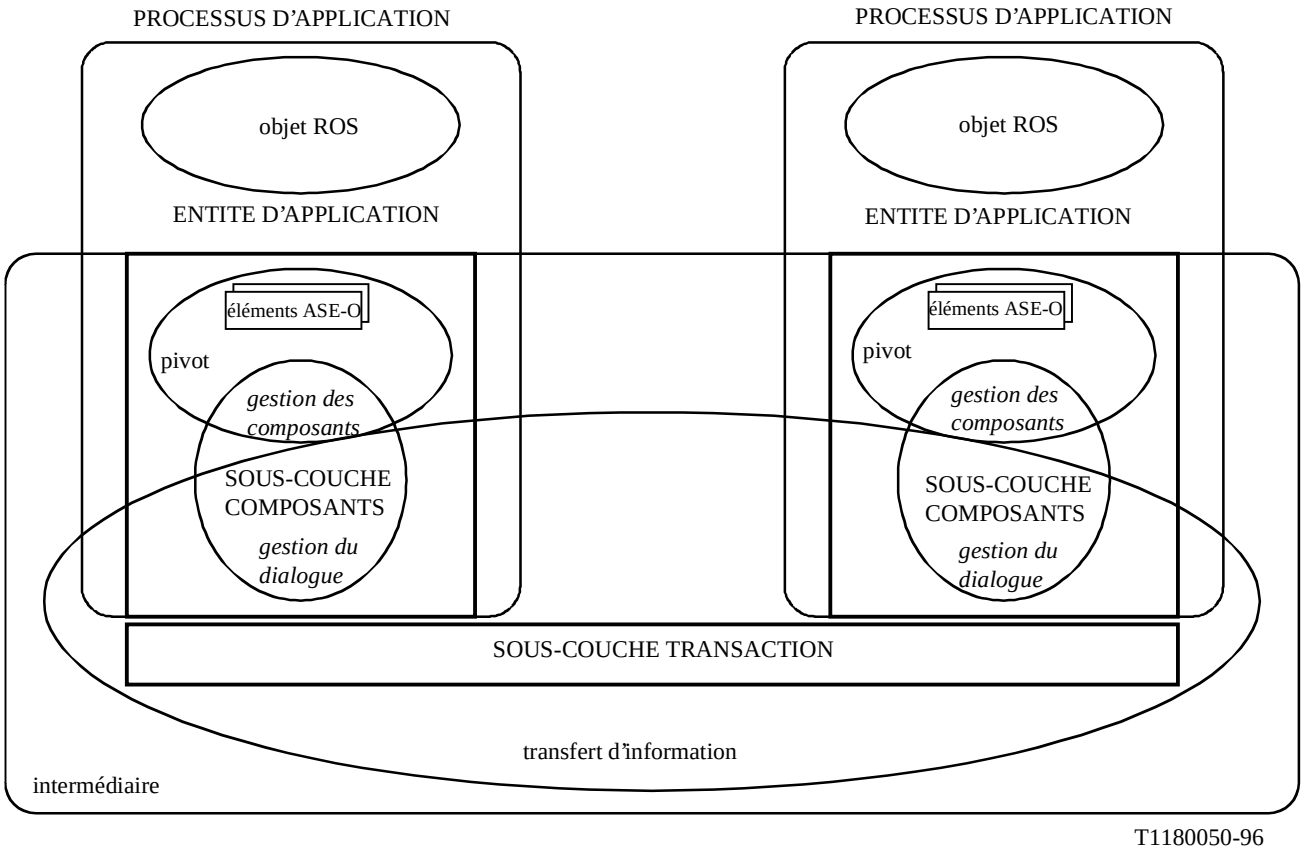


Figure 2/Q.775 – Réalisation du service ROS par le TC

5.2 Réalisation du service d'opérations distantes

5.2.1 Services de base (pivot)

La sous-couche composants du TC assure les services nécessaires à la prise en charge de l'invocation des opérations et de la notification des réponses. Elle assure aussi des services locaux supplémentaires d'annulation d'opérations (primitive de demande TC-U-CANCEL, primitive d'indication TC-L-CANCEL) ou de notification d'erreurs de protocole détectées localement (primitive d'indication TC-L-REJECT).

Il convient de noter les restrictions suivantes:

- l'ensemble des identificateurs d'invocation InvokeIds autorisés est limité à la gamme des nombres entiers (–128 à 127);
- il n'est pas tenu compte du champ &synchronous dans la définition d'une opération. Du point de vue d'un TC, les opérations sont toujours considérées comme étant asynchrones. Cela n'empêche toutefois pas l'utilisateur d'un TC de se comporter sur un mode synchrone;
- il n'est pas tenu compte des champs de priorité dans la définition d'une opération[1].

[1] Cette situation est susceptible d'évoluer à mesure que les études sur la gestion des priorités dans le cadre du système de signalisation n° 7 progresseront.

5.2.2 Opérations de rattachement Bind et de détachement Unbind

Le TC ne prévoit aucun mécanisme spécifique pour l'invocation des opérations de rattachement et de détachement. Il appartient à l'utilisateur du TC de construire les unités APDU de rattachement et de détachement et de les transférer au TC comme n'importe quel autre type d'information d'usager. Ignorant par conséquent quand se déclenche l'invocation de ces opérations, le TC ne vérifie pas si la manière dont celles-ci sont utilisées est compatible avec le service de dialogue et le service de gestion des composants (il lui est impossible de vérifier qu'aucune opération n'est demandée après l'invocation d'une opération de détachement).

5.2.2.1 Opération de rattachement

Lorsque la définition d'un contexte d'application inclut un lot de connexion, l'utilisateur du TC initiateur invoque l'exécution d'une opération de rattachement dans le cadre de la procédure d'établissement du dialogue, avant l'exécution de toute autre opération. L'échec de l'exécution de cette opération entraîne le rejet du dialogue.

Si l'utilisateur du TC n'a pas vraiment besoin d'invoquer une opération de rattachement explicite, il utilisera en principe l'opération prédéfinie de rattachement vide emptyBind.

5.2.2.1.1 Invocation d'une opération de rattachement

L'utilisateur du TC peut invoquer une opération de rattachement en utilisant une primitive de demande TC-BEGIN. Si la définition de l'opération de rattachement inclut un champ &ArgumentType, l'utilisateur du TC construit à partir de cette information une unité PDU d'invocation de rattachement qu'il transfère dans la première (ou seule) partie du paramètre d'information d'utilisateur de la primitive de demande TC-BEGIN. Si tel n'est pas le cas, aucune unité PDU d'invocation de rattachement n'est envoyée.

NOTE – Cette manière de procéder devrait permettre d'inclure l'unité PDU de demande de rattachement dans le premier champ externe de l'élément d'information d'utilisateur de l'unité APDU de demande de dialogue (AARQ).

5.2.2.1.2 Réponse à une opération de rattachement

L'utilisateur du TC notifie le résultat d'une opération de rattachement au moyen de la première primitive de gestion de dialogue qu'il émet.

La bonne exécution de l'opération de rattachement est notifiée au moyen d'une primitive de demande TC-CONTINUE ou d'une primitive de demande TC-END s'il n'est pas nécessaire de poursuivre le dialogue. Dans ce cas, il convient aussi d'invoquer une opération de détachement.

NOTE – L'utilisation de la primitive de demande TC-END à ce stade impose des restrictions à l'utilisation d'opérations de détachement. Elle suppose que seul le répondeur est habilité à exécuter une opération de détachement et que la définition d'une telle opération n'inclut pas de champ &ResultType et encore que la définition de l'erreur qui lui est associée n'inclut pas de champ &ParameterType (comme l'opération de détachement vide emptyUnbind par exemple).

Si la définition de l'opération de rattachement inclut un champ &ResultType, l'utilisateur du TC construit à partir de cette information une unité PDU de résultat de rattachement qu'il transfère dans la première (ou seule) partie du paramètre d'information d'utilisateur de la primitive de demande TC-CONTINUE ou de la primitive de demande TC-END. Si tel n'est pas le cas, aucune unité PDU de résultat de rattachement n'est envoyée.

L'utilisateur du TC notifie l'échec de l'exécution d'une opération de rattachement en émettant une primitive de demande TC-U-ABORT immédiatement en réponse à l'indication TC-BEGIN. Le paramètre indiquant la raison de l'abandon prend la valeur "dialogue refusé".

Si la définition de l'erreur associée inclut un champ &ParameterType, l'utilisateur du TC construit à partir de cette information une unité PDU d'erreur de rattachement qu'il transfère dans la première (ou seule) partie du paramètre d'information d'utilisateur de la primitive de demande TC-CONTINUE ou de la primitive de demande TC-END. Si tel n'est pas le cas, aucune unité PDU d'erreur de rattachement n'est envoyée.

L'opération de rattachement vide emptyBind et les unités PDU d'invocation de rattachement, de résultat de rattachement et d'erreur de rattachement sont définies dans la Recommandation X.880. Pour plus de commodité, leurs définitions en ASN.1 sont reproduites ci-dessous:

```
Bind {OPERATION:operation} ::= CHOICE
{
        bind-invoke             [16]    OPERATION.&ArgumentType (operation),
        bind-result             [17]    OPERATION.&ResultType (operation),
        bind-error              [18]    OPERATION.&Errors.&ParameterType (operation)
}

emptyBind OPERATION ::= {ERRORS {refuse} SYNCHRONOUS TRUE}
```

On entend ici par *opération* l'opération de rattachement.

5.2.2.2 Opération de détachement

5.2.2.2.1 Invocation d'une opération de détachement

Si la définition du contexte d'application inclut un lot de connexion, l'utilisateur du TC invoque une opération de détachement dans le cadre de la procédure mettant fin au dialogue.

Le mappage avec les services du TC dépend du type d'opération de détachement:

a) si la définition de l'opération de détachement n'inclut pas un champ &ResultType et que la définition de l'erreur qui lui est associée n'inclut pas un champ &ParameterType, l'opération peut être invoquée à l'aide de la primitive de demande TC-END.

b) Si la définition de l'opération de détachement inclut un champ &ResultType ou que la définition de l'erreur qui lui est associée inclut un champ &ParameterType, l'opération doit être invoquée au moyen de la dernière primitive de demande TC-CONTINUE émise par le demandeur du détachement.

Dans les deux cas, si la définition de l'opération de détachement inclut un champ &ArgumentType, l'utilisateur du TC construit une unité APDU de demande de détachement qu'il transfère dans la dernière (ou seule) partie du paramètre d'information d'utilisateur de la primitive de demande TC-END. Dans les autres cas, aucune unité APDU de demande de détachement n'est envoyée.

5.2.2.2.2 Réponse à une opération de détachement

Lorsqu'il accepte une opération de détachement, l'utilisateur du TC émet une primitive de demande TC-END. Si la définition de l'opération de détachement inclut un champ &ResultType, l'utilisateur du TC construit une unité APDU de résultat de détachement qu'il transfère dans la dernière (ou seule) partie du paramètre d'information d'utilisateur de la primitive de demande TC-END. Si tel n'est pas le cas, aucune unité APDU de résultat de détachement n'est envoyée.

Lorsqu'il refuse une opération de détachement, l'utilisateur du TC émet une primitive de demande TC-CONTINUE. Si la définition de l'erreur associée inclut un champ &ParameterType, l'utilisateur du TC construit une unité APDU d'erreur de détachement qu'il transfère dans la dernière (ou seule) partie du paramètre d'information d'utilisateur d'une primitive de demande TC-END. Si tel n'est pas le cas, aucune unité APDU de résultat de détachement n'est envoyée.

NOTE – Cette manière de procéder devrait permettre d'inclure l'unité PDU de résultat de détachement dans le dernier champ externe de l'élément d'information d'utilisateur de l'unité APDU de réponse de dialogue (AARE) en cas d'émission de la primitive de demande TC-END immédiatement en réponse à la primitive d'indication TC-BEGIN ou sinon dans le champ EXTERNAL unique de la portion de dialogue.

Si le contrat d'association inclut un lot de connexion mais que l'utilisateur du TC n'a pas besoin d'invoquer explicitement une opération de détachement, on part du principe que l'opération de détachement vide emptyUnbind est utilisée. Cette opération est théoriquement mappée à la primitive de demande TC-END, bien qu'aucune unité PDU de détachement ne soit envoyée.

L'opération de détachement vide emptyUnbind ainsi que les unités PDU d'invocation de détachement, de résultat de détachement et d'erreur de détachement sont définies dans la Recommandation X.880. Pour plus de commodité, elles sont reproduites ci-dessous:

```
Unbind {OPERATION:operation} ::= CHOICE
{
        unbind-invoke       [19]    OPERATION.&ArgumentType (operation),
        unbind-result       [20]    OPERATION.&ResultType (operation),
        unbind-error        [21]    OPERATION.&Errors.&ParameterType (operation)
}

emptyUnbind OPERATION ::= {SYNCHRONOUS TRUE}
```

On entend par *opération* l'opération de détachement.

5.3 Transfert d'information

5.3.1 Réalisations d'association

Le TC assure deux réalisations d'association par l'intermédiaire de sa fonction de gestion de dialogue: le mode structuré et le mode non structuré qui sont définis dans la Recommandation Q.771.

5.3.2 Réalisation de transfert

Pour ce qui est des opérations distantes, le TC offre à son utilisateur les capacités de transfert d'information suivantes:

- transfert des unités PDU de rattachement bind et de détachement unbind dans une information d'utilisateur à l'intérieur de la portion de dialogue;
- transfert des unités PDU ROS de base (plus retour du résultat partiel) dans la portion de composant de tout message.

Le TC n'assure qu'un type de réalisation de transfert, indépendamment du type de réalisation d'association choisi. Toutefois, du point de vue d'un expéditeur, cette réalisation offre une certaine latitude aux utilisateurs de TC en ce qui concerne la concaténation des unités PDU.

Outre les opérations distantes, le TC permet également de transférer n'importe quel type d'information d'utilisateur au moyen de primitives de service de gestion de dialogue.

5.4 Contexte d'application utilisant le TC

Les aspects statiques de la définition d'un contexte d'application utilisant le TC réalisant un contrat d'association particulier peuvent être décrits comme un objet informationnel de la classe APPLICATION-CONTEXT, spécifié comme suit:

```
APPLICATION-CONTEXT ::=                          CLASS
{
        &associationContract                     CONTRACT,
        &dialogueMode                            DialogueMode,
        &termination                             Termination OPTIONAL,
        &componentGrouping                       BOOLEAN DEFAULT TRUE,
        &dialogueAndComponentGrouping            BOOLEAN DEFAULT TRUE,
        &AdditionalASEs                          OBJECT IDENTIFIER OPTIONAL,
        &AbstractSyntaxes                        ABSTRACT-SYNTAX,
        &applicationContextName                  OBJECT IDENTIFIER UNIQUE
}

WITH SYNTAX
{
        CONTRACT                                 &associationContract
        DIALOGUE MODE                            &dialogueMode
        [TERMINATION                             &termination]
        [COMPONENT GROUPING ALLOWED              &componentGrouping]
        [DIALOGUE WITH COMPONENTS ALLOWED        &dialogueAndComponentGrouping]
        [ADDITIONAL ASEs                         &AdditionalASEs]
        ABSTRACT SYNTAXES                        &AbstractSyntaxes
        APPLICATION CONTEXT NAME                 &applicationContextName
}
DialogueMode ::= ENUMERATED {structured (1), unstructured (2)}

Termination ::= ENUMERATED {basic (1), prearranged (2)}
```

Le champ &associationContract identifie le contrat d'association que réalise ce contexte d'application.

Le champ &dialogueMode indique si ce contexte d'application utilise les fonctions du mode de dialogue structuré ou celles du mode de dialogue non structuré. Si la définition du contrat d'association inclut un lot de connexion, le champ &dialogueMode doit indiquer "structuré".

Le champ &termination indique si la terminaison de base ou la terminaison prédéterminée est utilisée pour mettre fin au dialogue. Si ce champ est absent, la définition du contexte d'application n'impose aucune contrainte quant au choix de la méthode à utiliser pour mettre fin au dialogue.

Le champ &componentGrouping indique si plusieurs composants peuvent être groupés dans un même message. Si ce champ est absent, la définition du contexte d'application n'impose pas de restrictions en la matière.

Le champ &dialogueAndComponentGrouping indique si les unités PDU de rattachement et de détachement peuvent être envoyées dans des messages qui contiennent aussi des composants. Si ce champ est absent, la définition du contexte d'application n'impose pas de restrictions à cet égard.

Le champ &AdditionalASEs contient les identificateurs d'objet des éléments ASE dont a besoin le contexte d'application (s'il en existe un) et qui ne font pas appel aux opérations distantes.

Le champ &AbstractSyntaxes contient les syntaxes abstraites nécessaires à l'acheminement de l'information entre les objets, y compris les unités PDU d'invocation et de notification des opérations du contrat.

Le champ &applicationContextName contient la valeur à fournir au TC pour identifier le contexte d'application.

5.5 Syntaxes abstraites

5.5.1 Commande de dialogue

Le champ &AbstractSyntaxes de la définition d'un contexte d'application doit inclure la syntaxe abstraite suivante si le champ &dialogueMode indique "structuré".

dialogue-abstract-syntax ABSTRACT-SYNTAX :: = {DialoguePDU IDENTIFIED BY dialogue-as-id}

Le champ &AbstractSyntaxes de la définition d'un contexte d'application doit inclure la syntaxe abstraite suivante si le champ &dialogueMode indique "non structuré".

uniDialogue-abstract-syntax ABSTRACT-SYNTAX :: = {UniDialoguePDU IDENTIFIED BY uniDialogue-as-id}

5.5.2 Syntaxes définies par l'utilisateur

5.5.2.1 Généralités

Le champ &AbstractSyntaxes de la définition d'un contexte d'application doit inclure une ou plusieurs syntaxes abstraites pour représenter les messages du TC (y compris les composants) et les unités PDU de rattachement et de détachement. Ces syntaxes abstraites doivent être définies par le concepteur de l'application.

Les messages du TC sont définis dans la Recommandation Q.773 tandis que les unités PDU de rattachement bind et de détachement unbind sont définies dans la Recommandation X.880.

Il appartient au concepteur de l'application de décider du nombre de syntaxes abstraites à définir pour un contexte d'application donné. Il convient toutefois de suivre les règles suivantes:

a) si le contexte d'application réalise un contrat d'association incluant un lot de connexion, les valeurs des types de données suivants:

Bind{ac.&associationContract.&connection.&bind}
Unbind{ac.&associationContract.&connection.&unbind}

doivent apparaître dans au moins une de ces syntaxes abstraites.

b) pour chaque opération *op* intervenant dans l'ensemble de lots d'opérations qu'utilise le contexte d'application, une des syntaxes abstraites au moins doit inclure les valeurs des types suivants:

Invoke {TCInvokeIds, OPERATION:op}
ReturnResult {OPERATION:op}

c) pour chaque erreur *err* intervenant dans l'ensemble de lots d'opérations qu'utilise le contexte d'application, une des syntaxes abstraites au moins doit inclure les valeurs des types suivants:

ReturnError {ERROR:err}

d) une des syntaxes abstraites au moins doit inclure:

Reject

5.5.2.2 Définition des syntaxes abstraites

Pour un lot d'opérations donné, une même syntaxe abstraite permettant l'échange de messages TC d'invocation et de notification pour toutes ses opérations peut être définie à l'aide du type de données suivant:

```
TCSingleAS {OPERATION-PACKAGE: package} :: = TCMessage { {AllOperations {package}},
{AllOperations {package}} }
```

On peut aussi définir une paire de syntaxes abstraites d'après la paire de types:

```
TCConsumerAS {OPERATION-PACKAGE: package} :: = TCMessage { {ConsumerPerforms {package}},
{ConsumerPerforms {package}} }

TCSupplierAS {OPERATION-PACKAGE: package} :: = TCMessage { {SupplierPerforms {package}},
{SupplierPerforms {package}} }
```

Une même syntaxe abstraite peut prendre en charge un ensemble de lots, à condition que les codes d'opération et d'erreur soient uniques. Par exemple, le type de données suivant peut être utilisé dans le cadre d'une même syntaxe abstraite pour prendre en charge tous les lots d'opérations intervenant dans un contrat d'association:

```
AllPackagesAS {APPLICATION-CONTEXT:ac} ::=
        TCSingleAS
        {
            combine
                {
                    {
                        ac.&associationContract.&OperationsOf
                        | ac.&associationContract.&InitiatorConsumerOf
                        | ac.&associationContract.&InitiatorSupplierOf
                    }
                    {},
                    {}
                }
        }
```

Une syntaxe abstraite indépendante peut être définie pour représenter les valeurs des unités PDU de rattachement bind et de détachement unbind, d'après le type suivant:

```
ConnectionAS {APPLICATION-CONTEXT:ac} ::= CHOICE
{
    bind        Bind{ac.&associationContract.&connection.&bind},
    unbind      Unbind{ac.&associationContract.&connection.&unbind}
}
```

La valeur de l'identificateur d'objet attribuée à cette syntaxe abstraite doit être incluse dans le champ multivaleur du nom de syntaxe abstraite &abstract-syntax-name de la définition du contexte d'application. Cette valeur est destinée à servir de référence directe lorsque les valeurs de ces unités PDU sont véhiculées dans le paramètre d'information d'utilisateur dans le cas d'unités PDU de commande de dialogue, ou directement comme valeur de la portion de dialogue.

D'autres syntaxes abstraites peuvent aussi être définies pour représenter les valeurs des unités PDU associées à des éléments ASE de type autre que ROS (voir l'exemple donné au 4.7).

5.6 Extension de notation

Le module ASN.1 suivant contient des définitions qui permettent au concepteur d'un protocole d'utilisateur de TC de spécifier des contextes d'application et des syntaxes abstraites en tant qu'instances de classes d'objets informationnels appropriées.

```
TC-Notation-Extensions   {ccitt recommendation q 775 modules(2) notation-extension(4) version1(1)}

DEFINITIONS ::=

BEGIN

IMPORTS

TCMessage{} FROM TCAPMessages {ccitt recommendation q 773 modules(2) messages(1) version3(3)}

Bind{}, Unbind{} FROM Remote-Operations-Generic-ROS-PDUs {joint-iso-ccitt remote-operations(4) generic-
ROS-PDUs(6) version1(0)}

AllOperations{}, ConsumerPerforms{}, SupplierPerforms{}, combine{} FROM Remote-Operations-Useful-
Definitions {joint-iso-ccitt remote-operations(4) useful-definitions(7) version1(0)}

CONTRACT, OPERATION-PACKAGE FROM Remote-Operations-Information-Objects {joint-iso-ccitt remote-
operations(4) informationObjects(5) version1(0)}

UniDialoguePDU, uniDialogue-as-id FROM UnidialoguePDUs
        {ccitt recommendation q 773 modules(2) unidialoguePDUs(3) version1(1)}

DialoguePDU, dialogue-as-id FROM DialoguePDUs
        {ccitt recommendation q 773 modules(2) dialoguePDUs(2) version1(1)}

APPLICATION-CONTEXT ::=                         CLASS
{
        &associationContract                    CONTRACT,
        &dialogueMode                           DialogueMode,
        &termination                            Termination OPTIONAL,
        &componentGrouping                      BOOLEAN DEFAULT TRUE,
        &dialogueAndComponentGrouping           BOOLEAN DEFAULT TRUE,
        &AdditionalASEs                         OBJECT IDENTIFIER OPTIONAL,
        &AbstractSyntaxes                       ABSTRACT-SYNTAX,
        &applicationContextName                 OBJECT IDENTIFIER UNIQUE
}

WITH SYNTAX
{
        CONTRACT                                &associationContract
        DIALOGUE MODE                           &dialogueMode
        [TERMINATION                            &termination]
        [COMPONENT GROUPING ALLOWED             &componentGrouping]
        [DIALOGUE WITH COMPONENTS ALLOWED       &dialogueAndComponentGrouping]
        [ADDITIONAL ASEs                        &AdditionalASEs]
        ABSTRACT SYNTAXES                       &AbstractSyntaxes
        APPLICATION CONTEXT NAME                &applicationContextName
}

DialogueMode ::= ENUMERATED {structured (1), unstructured (2)}

Termination ::= ENUMERATED {basic (1), prearranged (2)}

dialogue-abstract-syntax ABSTRACT-SYNTAX ::= {DialoguePDU IDENTIFIED BY dialogue-as-id}

uniDialogue-abstract-syntax ABSTRACT-SYNTAX ::= {UniDialoguePDU IDENTIFIED BY uniDialogue-as-id

TCSingleAS {OPERATION-PACKAGE: package} ::= TCMessage { {AllOperations {package}}, {AllOperations
{package}} }
```

```
TCConsumerAS {OPERATION-PACKAGE: package} ::= TCMessage { {ConsumerPerforms {package}},
{ConsumerPerforms {package}} }

TCSupplierAS {OPERATION-PACKAGE: package} ::= TCMessage { {SupplierPerforms {package}},
{SupplierPerforms {package}} }

AllPackagesAS {APPLICATION-CONTEXT:ac} ::=
        TCSingleAS
                {
                        combine
                                {
                                        {
                                                ac.&associationContract.&OperationsOf
                                                | ac.&associationContract.&InitiatorConsumerOf
                                                | ac.&associationContract.&InitiatorSupplierOf
                                        },
                                        {},
                                        {}
                                }
                }
ConnectionAS {APPLICATION-CONTEXT:ac} ::= CHOICE
{
        bind            Bind{ac.&associationContract.&connection.&bind},
        unbind          Unbind{ac.&associationContract.&connection.&unbind}
}
END
```

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