UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Q.834.4

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(07/2003)

SÉRIE Q: COMMUTATION ET SIGNALISATION

Interface Q3

Spécification d'interface en architecture CORBA pour les réseaux optiques passifs à large bande basée sur les prescriptions d'interface UML

Recommandation UIT-T Q.834.4

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, 5, 6, R1 ET R2	Q.120–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.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.1699
PRESCRIPTIONS ET PROTOCOLES DE SIGNALISATION POUR LES IMT-2000	Q.1700–Q.1799
SPÉCIFICATIONS DE LA SIGNALISATION RELATIVE À LA COMMANDE D'APPEL INDÉPENDANTE DU SUPPORT	Q.1900–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.834.4

Spécification d'interface en architecture CORBA pour les réseaux optiques passifs à large bande basée sur les prescriptions d'interface UML

Résumé

La présente Recommandation donne une définition du langage IDL d'architecture CORBA pour l'interface de gestion entre un système de gestion de fournisseur et un système de gestion d'opérateur. Elle définit une partie des aspects de gestion pour les ressources de réseau définies dans la série de Recommandations UIT-T G.983.x pour l'équipement de réseau optique passif à large bande (BPON, *broadband passive optical network*).

D'une façon générale, le système de gestion par fournisseur est un système de gestion d'éléments (EMS, *element management system*) et le système de gestion par opérateur (OMS, *operator management system*) est un système de gestion de réseau (NMS, *network management system*). Cependant, le système de gestion par fournisseur est tenu de présenter au système de gestion par opérateur une "vue réseau" de la gestion des connexions. Il a donc été jugé nécessaire, par souci de clarté, d'utiliser la terminologie adoptée afin de désigner les systèmes concernés.

En outre, il convient de noter que la Rec. UIT-T Q.834.1 contient un ensemble de prescriptions relatives aux fonctionnalités et une liste des définitions d'entité gérée constituant la base des informations de gestion requises pour une "vue par élément de réseau" de l'équipement de réseau BPON. La Rec. UIT-T Q.834.2 complète la définition des informations de gestion pour la gestion de l'équipement de réseau BPON en fournissant les définitions des entités gérées pour la "vue réseau". La Rec. UIT-T Q.834.3 traite du comportement de l'interface de gestion au moyen de diagrammes en langage UML et de descriptions de scénario. Les informations de gestion modélisées dans les Recommandations UIT-T Q.834.1 et Q.834.2 sont citées en référence dans la totalité de la présente Recommandation et de la Rec. UIT-T Q.834.3.

Source

La Recommandation Q.834.4 de l'UIT-T a été approuvée le 7 juillet 2003 par la Commission d'études 4 (2001-2004) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8. Cette édition inclut les modifications introduites par le Corrigendum 1 approuvé le 13 janvier 2004.

Mots clés (facultatifs)

APON, BPON, CORBA, IDL, PON, UML.

AVANT-PROPOS

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

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

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

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

NOTE

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

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

DROITS DE PROPRIÉTÉ INTELLECTUELLE

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

A la date d'approbation de la présente Recommandation, l'UIT 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	Domaine d'application	1
2	Références	1
	2.1 Références normatives	1
	2.2 Autres références	2
3	Termes et définitions	2
	3.1 Termes importés de M.3010	2
	3.2 Termes importés du langage UML	3
	3.3 Termes importés du service de nommage du groupe OMG	3
	3.4 Termes importés de la Rec. UIT-T Q.834.1	3
	3.5 Termes importés de la Rec. UIT-T Q.834.3	3
	3.6 Nouveaux termes	3
4	Abréviations	4
5	Conventions	5
	5.1 Conventions relatives à la description des modules	5
	5.2 Conventions relatives aux fichiers en langage IDL	6
	5.3 Valeurs vides	6
6	Aperçu général de l'architecture d'interface	6
7	Noms et contrainte de nommage	7
	7.1 Objets de service et service de nommage du groupe OMG	8
	7.2 Objets de domaine	9
8	Organisation des fichiers en langage IDL	10
9	Modules	11
	9.1 Module AccessControl	11
	9.2 Module Build	17
	9.3 Module Q834Common	27
	9.4 Module ControlArchive	28
	9.5 Module SoftwareDownload	32
	9.6 Module EventPublisher	40
	9.7 Module MIBTransfer	43
	9.8 Module PerformanceManager	47
	9.9 Module ProfileManager	56
	9.10 Module Registrar	58
	9.11 Module ResourceAllocation	63
	9.12 Module SchedulerManagement	67
	9.13 Module ServiceProvisioning	71
	9.14 Module Synchroniser	75
	9.15 Module Test	78
	9.16 Module FileTransfer	85

Page

10 Déclaration de conformité 88

Annexe A – Dictionnaire de données 88

Annexe B – Exceptions 114

Annexe C – Fichiers en langage IDL 120

C.1 Q834AccessControl.idl 120

C.2 Q834Build.idl 125

C.3 Q834Common.idl 131

C.4 Q834ControlArchive.idl 145

C.5 Q834SoftwareDownload.idl 148

C.6 Q834EventPublisher.idl 153

C.7 Q834MIBTransfer.idl 157

C.8 Q834PerformanceManager.idl 160

C.9 Q834ProfileManager.idl 165

C.10 Q834Registrar.idl 176

C.11 Q834ResourceAllocation.idl 179

C.12 Q834SchedulerManagement.idl 182

C.13 Q834ServiceProvisioning.idl 185

C.14 Q834Synchroniser.idl 187

C.15 Q834Test.idl 190

C.16 Q834Filetransfer.idl 195

Annexe D – Exemple de gabarits d'extrémité 198

Recommandation UIT-T Q.834.4

Spécification d'interface en architecture CORBA pour les réseaux optiques passifs à large bande basée sur les prescriptions d'interface UML

1 Domaine d'application

La présente Recommandation traite de la conception d'une interface interactive mécanisée entre d'une part le système de gestion par fournisseur qui gère des ressources de réseau BPON et un système de gestion par opérateur (OMS). La conception est fondée sur les diagrammes en langage UML et sur les descriptions de scénario de la Rec. UIT-T Q.834.3. L'approche générale adoptée dans la Rec. UIT-T Q.834.3 et implémentée ici est telle que le système de gestion par fournisseur fournit des services de gestion au système de gestion par opérateur. Ces services donnent au système OMS une abstraction de haut niveau des capacités suivantes:

- fourniture de ressources de réseau installées;
- fourniture de ressources de réseau non installées, y compris la réservation de capacité;
- fourniture de services;
- gestion des archives;
- gestion des logiciels d'élément de réseau;
- sauvegarde et restauration des données de configuration d'élément de réseau;
- gestion de la qualité de fonctionnement;
- publication des événements d'élément de réseau;
- gestion des profils;
- essais;
- programmation des activités;
- gestion des transferts en masse;
- synchronisation NE-EMS;
- contrôle d'accès.

La présente Recommandation décrit les interfaces en langage IDL de l'architecture CORBA prenant en charge les services énumérés ci-dessus.

2 Références

2.1 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui, de ce fait, en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée. La référence à un document figurant dans la présente Recommandation ne donne pas à ce document, en tant que tel, le statut d'une Recommandation.

[1] Recommandation UIT-T M.3010 (2000), *Principes du réseau de gestion des télécommunications*.

[2] Recommandation UIT-T M.3200 (1997), *Services de gestion du réseau de gestion des télécommunications et domaines gérés des télécommunications: aperçu général*.

[3] Recommandation UIT-T M.3400 (2000), *Fonctions de gestion du réseau de gestion des télécommunications*.

[4] Document OMG formel/99-06-01, *Langage de modélisation unifié, Section 1*.

[5] Recommandation UIT-T G.983.1 (1998), *Systèmes d'accès optique à large bande basés sur un réseau optique passif*, plus Amendement 1 (2001).

[6] Recommandation UIT-T Q.834.1 (2001), *Prescriptions et entités gérées d'un réseau optique passif ATM pour la vue d'élément de réseau*.

[7] Recommandation UIT-T Q.834.2 (2001), *Prescriptions et entités gérées d'un réseau optique passif ATM pour la vue de réseau*.

[8] Recommandation UIT-T Q.834.3 (2001), *Description en langage UML des prescriptions relatives aux interfaces de gestion des réseaux optiques passifs à large bande*.

[9] Recommandation UIT-T M.3020 (2000), *Méthodologie pour la spécification des interfaces du réseau de gestion des télécommunications*.

[10] Recommandation UIT-T G.983.2 (2002), *Spécification de l'interface de gestion et de commande de terminaison de réseau optique pour réseau optique passif à large bande*.

[11] Recommandation UIT-T G.983.3 (2001), *Système d'accès optique à large bande avec capacité de service accrue par attribution de longueur d'onde*.

[12] Recommandation UIT-T G.983.4 (2001), *Système d'accès optique à large bande avec capacité de service accrue par assignation dynamique de largeur de bande*.

[13] Recommandation UIT-T G.983.5 (2002), *Système d'accès optique à large bande avec capacité de survie améliorée*.

[14] Recommandation UIT-T G.983.6 (2002), *Spécifications de l'interface de gestion et de commande des terminaisons de réseau optique pour les réseaux optiques passifs à large bande à dispositifs de protection*.

[15] Recommandation UIT-T G.983.7 (2001), *Spécification de l'interface de gestion et de commande de terminaison optique pour système de réseau optique passif à large bande avec attribution dynamique de largeur de bande*.

[16] Recommandation UIT-T X.780 (2001), *Directives concernant le RGT pour la définition d'objets gérés CORBA*.

[17] Recommandation UIT-T Q.816 (2001), *Services RGT à architecture CORBA*.

2.2 Autres références

[18] Groupe de gestion d'objets (OMG), "CORBA Services – *Naming Service Specification*", OMG Document formal/02-09-02, février 2002.

[19] Groupe de gestion d'objets (OMG), "CORBA Services – *Notification Service*", OMG Document formal/02-08-04, août 2002.

3 Termes et définitions

3.1 Termes importés de M.3010

Le terme suivant de la Rec UIT-T M.3010 est utilisé dans la présente Recommandation:

– utilisateur.

3.2 Termes importés du langage UML

Les termes suivants du langage UML [4] sont utilisés dans la présente Recommandation:

– acteur;
– classe;
– diagramme de classe;
– scénario.

3.3 Termes importés du service de nommage du groupe OMG

Les termes suivants du service de nommage du groupe OMG [18] sont utilisés dans la présente Recommandation:

– graphe de nommage;
– nom.

3.4 Termes importés de la Rec. UIT-T Q.834.1

Le terme suivant de la Rec. UIT-T Q.834.1 est utilisé dans la présente Recommandation:

– entité gérée.

3.5 Termes importés de la Rec. UIT-T Q.834.3

Les termes suivants de la Rec. UIT-T Q.834.3 sont utilisés dans la présente Recommandation:

– activation;
– assignation;
– autodécouverte;
– ressource de réseau BPON;
– construction;
– filtrage;
– installation;
– télémétrie;
– inscription;
– réservation;
– instance de service;
– étiquette d'utilisateur.

3.6 Nouveaux termes

La présente Recommandation définit les termes suivants:

3.6.1 objet de service: ensemble des objets qui donnent accès aux services de gestion implémentés dans le système de gestion par fournisseur. Les interfaces avec des objets de service constituent la spécification pour la portion de l'interface IF1 qui est définie dans la présente Recommandation.

3.6.2 objet de domaine: ensemble des objets qui définissent les informations de gestion pour la gestion des ressources de réseau BPON. Les objets de domaine sont principalement fournis par les listes d'entités gérées des Recommandations UIT-T Q.834.1 et Q.834.2.

3.6.3 objet interne: ensemble des objets qui servent à prendre en charge la logique interne d'un système de gestion par fournisseur. Ces objets sont complètement obscurs du point de vue du système OMS.

4 Abréviations

La présente Recommandation utilise les abréviations suivantes:

AAL couche d'adaptation au mode ATM (*ATM adaptation layer*)

APON réseau optique passif en mode ATM (*ATM passive optical network*)

ATM mode de transfert asynchrone (*asynchronous transfer mode*)

BICI interface entre exploitants à large bande (*broadband inter-carrier interface*)

BISSI interface entre systèmes de commutation à large bande (*broadband inter-switching system interface*)

BPON réseau optique passif à large bande (*broadband passive optical network*)

CAC contrôle d'admission d'appel (*call admission control*)

CCITT Comité Consultatif International Télégraphique et Téléphonique

CES service d'émulation de circuit (*circuit emulation service*)

CNM gestion de réseau client (*customer network management*)

CORBA architecture de courtier commun de requête d'objets (*common object request broker architecture*)

CTP point de terminaison de connexion (*connection termination point*)

CTT ticket de panne client (*customer trouble ticket)*

DSx signal numérique x (*digital signal x*)

EM gestion d'élément (*element management*)

EML couche de gestion d'élément (*element management layer*)

EMS système de gestion d'élément (*element management system*)

EOC canal d'exploitation intégré (*embedded operations channel*)

Ex signal numérique européen x (*European digital signal x*)

GUI interface utilisateur graphique (*graphical user interface*)

IDL langage de définition d'interface (*interface definition language*)

ME entité gérée (*managed entity*)

MIB base d'informations de gestion (*management information base*)

NE élément de réseau (*network element*)

NMS système de gestion de réseau (*network management system*)

NT terminal de réseau (*network terminal*)

ODN réseau de distribution optique (*optical distribution network*)

OLT terminal de ligne optique (*optical line terminal*)

OMG groupe de gestion d'objets (*object management group*)

OMS système de gestion par opérateur (*operator management system*)

ONT terminal de réseau optique (*optical network terminal*)

ONU unité optique de réseau (*optical network unit*)

OS système d'exploitation (*operations system*)

PON réseau optique passif (*passive optical network*)

PVC circuit virtuel permanent (*permanent virtual circuit*)

QS qualité de service

RASC réseau d'accès à services complets

RCD réseau de communication de données

RGT réseau de gestion des télécommunications

TCA alerte de dépassement de seuil (*threshold crossing alert*)

TP point de terminaison (*termination point*)

TTP point de terminaison de chemin (*trail termination point*)

UIT Union internationale des télécommunications

UML langage de modélisation unifié (*unified modelling language*)

UNI interface utilisateur-réseau (*user-network interface*)

VC voie virtuelle (*virtual channel*)

VCC connexion par voie virtuelle (*virtual channel connection*)

VCI identificateur de voie virtuelle (*virtual channel identifier*)

VP conduit virtuel (*virtual path*)

VPC connexion par conduit virtuel (*virtual path connection*)

VPI identificateur de conduit virtuel (*virtual path identifier*)

5 Conventions

Des conventions spécifiques ont été suivies dans deux paragraphes de la présente Recommandation suivent. Le premier (§ 8) décrit les modules de langage IDL et le second (Annexe C) contient les fichiers en langage IDL. Finalement, la présente Recommandation utilise certaines conventions qui mettent en jeu des valeurs vides.

5.1 Conventions relatives à la description des modules

Les descriptions de module ont la structure suivante:

– Nom de module – vue d'ensemble des services, y compris toute modélisation à haut niveau, toutes descriptions clés d'éléments de données d'entreprise et tous scénarios, selon les besoins et l'applicabilité.

  • Nom d'interface – vue d'ensemble des services spécifiques fournis par cette interface.

    – Nom_1 d'opération – description détaillée du comportement de cette opération y compris la signature de l'opération, la description de chaque paramètre d'entrée, et la description de la valeur de retour.

    – …

    – Nom_n d'opération – description détaillée du comportement de cette opération y compris la signature de l'opération, la description de chaque paramètre d'entrée et la description de la valeur de retour.

    – Exceptions – conditions propres au contexte qui provoquent le déclenchement des exceptions nommées.

5.2 Conventions relatives aux fichiers en langage IDL

Chaque fichier en langage IDL a le format suivant:

- **#ifndef __<MODULENAME>_DEFINED** (le nom de module est le même que le nom de fichier)
- **#define __<MODULENAME>_DEFINED**
- **#include "<idlfilename>";**[1]
- **#pragma prefix "itu.Int"**
- **module q834_4 {**
- **module <modulename> {**
- toutes les définitions de type de données spécifiques et toutes les spécifications d'interface requises et indiquées par la conception d'interface issue de l'examen de la Rec. UIT-T Q.834.3;
- **}; //module <modulename>**
- **}; //module q834_4**

A l'intérieur de chaque définition de module, les définitions de type de données précèdent les définitions d'interface et sont segmentées par elles. Tous les types de données définis dans la présente Recommandation commencent par un caractère alphabétique initial qui est mis en majuscule. Si des noms de type de données sont des associations de plusieurs noms propres, pronoms, ou adjectifs, chaque association devra avoir un caractère alphabétique en majuscule au début de chaque composant. Les étiquettes des paramètres utilisés pour toute signature d'opération d'interface commencent par un caractère alphabétique en minuscule. Chaque nom d'opération commence par un caractère alphabétique en minuscule.

5.3 Valeurs vides

Dans la présente Recommandation, lorsqu'il est question d'une "chaîne vide", il s'agit d'une chaîne vide et non pas d'un objet vide pris en charge par des langages tels que JAVA. De même, lorsqu'il est question d'une "séquence vide", une séquence avec zéro élément est transmise.

6 Aperçu général de l'architecture d'interface

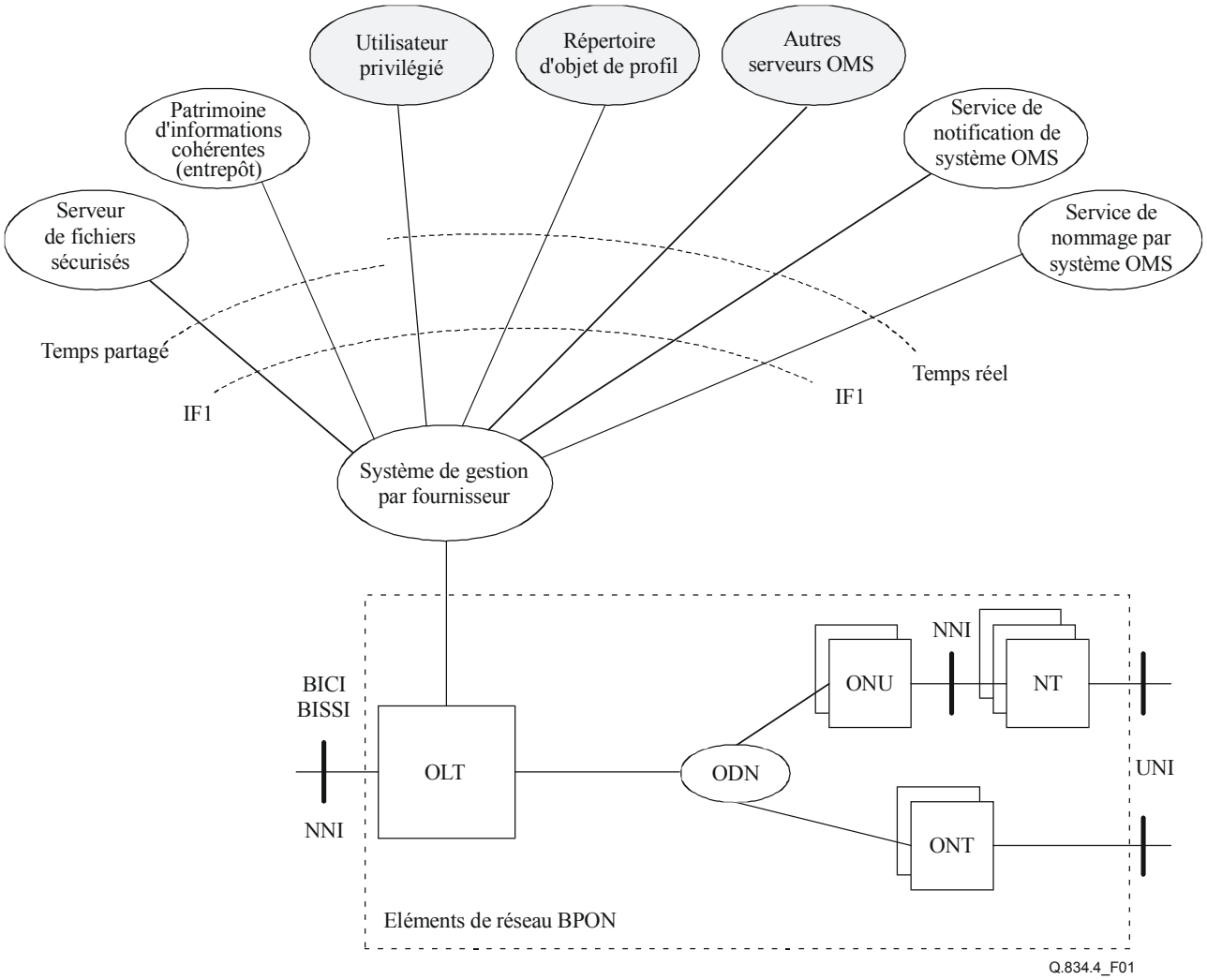
La Figure 1 décrit l'architecture et le système de gestion du réseau BPON. Cette technique offre un mécanisme intégré d'accès local pour chaque service de télécommunication actuellement déployé par des opérateurs. Une liste de tels services comprend les services de téléphonie et à fréquences vocales, les services d'émulation de circuit, les services Ethernet, ATM, xDSL et les services vidéo. Cet effort de normalisation étudie actuellement des solutions de transport dans les couches 1 et 2 du réseau ODN, bien que des implémentations initiales existent dans des réseaux d'opérateur mettant en jeu des réseaux APON et ATM comme définis dans la Rec. UIT-T G.983.1.

Cette figure montre également l'interface IF1 (Q) entre les systèmes de gestion par fournisseur et par opérateur. L'interface IF1 couvre tous les aspects de la gestion de la fourniture du réseau et des services, de la gestion de la qualité de fonctionnement du réseau, de la gestion du trafic, de la maintenance, des essais et de l'administration de la sécurité par l'utilisateur. La spécification de l'interface IF1 aurait été une tâche écrasante si la méthode adoptée n'avait pas recherché un niveau d'abstraction situé au-dessus des détails relatifs à la technique d'acheminement et de transport ainsi que des complexités propres à la gestion d'un aussi grand nombre de types de service. La méthode suivie a inclus la définition et la description des exigences d'interface au moyen du langage de

[1] Le fichier Q834Common.idl est inclus dans la présente Recommandation avec un fichier IDL sur deux.

modélisation unifié, a suivi la méthode de la Rec. UIT-T M.3020, et a été décrite dans la Rec. UIT-T Q.834.3.

Les systèmes de gestion par opérateur décrits dans ce diagramme correspondent directement aux acteurs de système définis dans la Rec. UIT-T Q.834.3. La présente Recommandation spécifie les transactions interactives en temps réel mettant en jeu le système de gestion par fournisseur et ne décrit pas explicitement les structures de fichier transférées entre le système de gestion par fournisseur et soit le patrimoine d'informations cohérentes (entrepôt de données) ou le serveur de fichiers sécurisés. Etant donné que les interfaces avec les services de nommage et de notification du groupe OMG sont déjà spécifiées, l'objet de la présente Recommandation se réduit à la messagerie entre le système de gestion par fournisseur et les acteurs de système sélectionnés.



BICI	interface entre exploitants à large bande	ODN	réseau de distribution optique
BISSI	interface entre systèmes de commutation à large bande	OLT	terminal de ligne optique
BPON	réseau optique passif à large bande	ONT	terminal de réseau optique
NNI	interface réseau-réseau	ONU	unité optique de réseau
NT	terminal de réseau	UNI	interface utilisateur-réseau

Figure 1/Q.834.4 – Vue d'ensemble de l'architecture d'interface

7 Noms et contrainte de nommage

Le présent paragraphe donne des lignes directrices relatives aux noms d'objet et aux identificateurs d'entité gérée cités en référence à l'interface IF1.

Une revue de tous les diagrammes de classe développés dans la Rec. UIT-T Q.834.3 montre que le système de gestion par fournisseur gère trois types distincts "d'objets". Ces types peuvent être classés comme suit:

- objets de service: ces objets donnent accès aux services de gestion implémentés dans le système de gestion par fournisseur. Les interfaces de ces services sont le principal sujet de la présente Recommandation. Les interfaces suivantes: NERegistrar, ServiceProvisioner, TestActionPerformer, et DownloadMgr en sont quelques exemples qui sont alignés sur la Rec. UIT-T X.780 en tant que dérivés de l'interface avec les objets gérés décrite dans la présente Recommandation;
- objets de domaine: ces objets définissent les informations de gestion des ressources de réseau BPON. Les objets de domaine sont principalement des ressources (entités gérées) désignées par les Recommandations UIT-T Q.834.1 et Q.834.2. Ces objets de domaine, ainsi que d'autres objets définis dans la Rec. UIT-T M.3120 et dans d'autres Recommandations conformes au cadre d'architecture CORBA de l'UIT-T, peuvent être mis à disposition pour utilisation à l'interface IF1. Un mécanisme pour harmoniser l'utilisation d'objets de service et d'objets de domaine sera expliqué dans la suite du présent paragraphe;
- objets internes: ces objets servent à prendre en charge un système de gestion par fournisseur logique interne. Ces objets sont complètement obscurs du point de vue du système OMS.

7.1 Objets de service et service de nommage du groupe OMG

Tous les objets de service (instanciés soit par le système de gestion du fournisseur soit par le système OMS) ont leur interface en langage IDL définie dans la présente Recommandation et leur référence d'objet inscrite auprès du service de nommage du groupe OMG décrit dans la Figure 1 au moyen de l'opération bind() de l'interface avec le service de nommage. Ultérieurement, l'emplacement de l'instance d'objet pourra être trouvé au moyen de l'opération resolve() du service de nommage, si nécessaire.

Les noms d'objet peuvent être soit "déjà identifiés" (c'est-à-dire que le nom a été décrit et accepté avant l'instant de l'exécution) ou communiqué à l'instant de l'exécution au moyen d'une autre interface. Dans le cas de la présente Recommandation, tous les objets de service sont "déjà identifiés". Le nom d'un objet déjà identifié peut être décrit au moyen d'un graphe de nommage. La Figure 2 donne un exemple de graphe de nommage pour certains des objets de service décrits dans la présente Recommandation, conformément à la convention de nommage décrite dans la Rec. UIT-T Q.816. Le diagramme représente l'identificateur suivi par un point suivi par la valeur de sorte. Seuls quatre des objets de service sont représentés par souci de simplicité dans la figure. Les autres objets de service seront nommés de manière analogue.

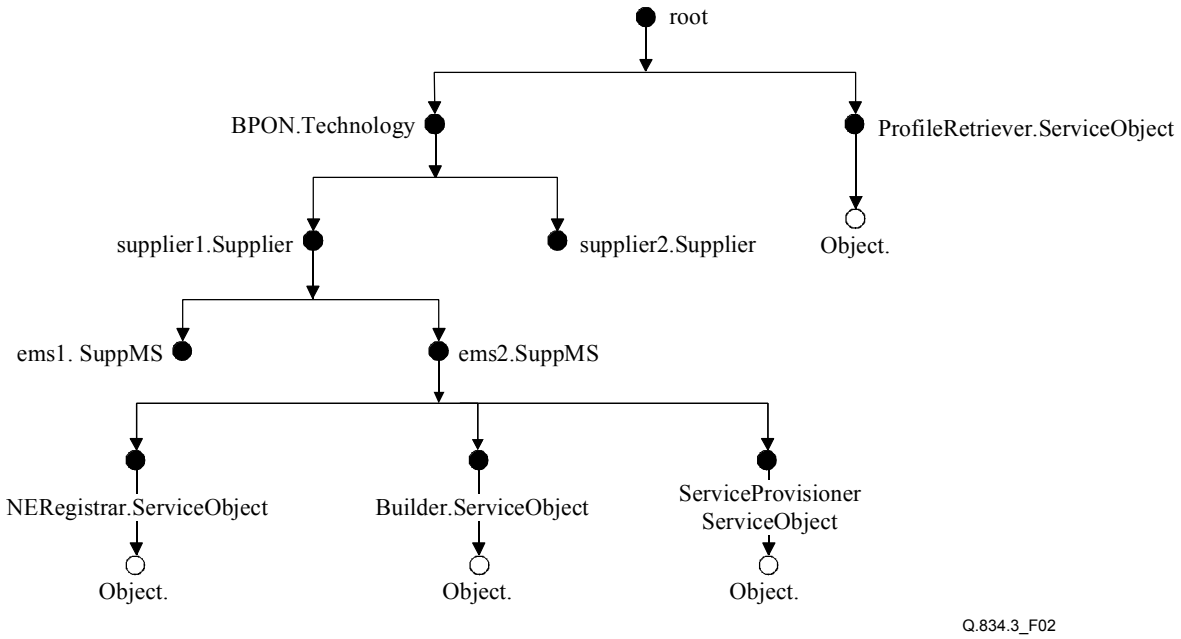


Figure 2/Q.834.4 – Graphe de nommage

Le chemin complet du nom est requis pour un objet "déjà identifié". Au démarrage d'un système de gestion par fournisseur nouvellement installé, toutes les instances d'objets de service sont automatiquement inscrites auprès d'un unique service de nommage du groupe OMG inscrit par l'opérateur. Le nom de tout objet inscrit auprès du service de nommage du groupe OMG a la syntaxe suivante:

```
typedef     string      Istring;

struct NameComponent {
       Istring id;
       Istring kind;
};

typedef sequence<NameComponent> Name;
```

Une interprétation du graphe de nommage de la Figure 2 est que le champ "sorte", contenu dans chacun des composants de nommage des objets de service a la valeur de chaîne vide et que chacun des nœuds représentés représente les valeurs pour le champ "id". Les détails du graphe de nommage ainsi que son interprétation devraient être déterminés au moyen d'un accord entre fournisseur et opérateur. Ces détails sont hors du domaine d'application de la présente Recommandation.

7.2 Objets de domaine

Le système de gestion par fournisseur conserve un grand nombre d'entités gérées (ou d'objets de domaine)[2]. Ces entités sont gérées indirectement au moyen des services de gestion offerts par le système de gestion du fournisseur. Cependant, les références (identificateurs) à des instances de ressources spécifiques sont souvent requises par le système OMS afin d'utiliser efficacement les services du système de gestion par fournisseur. Quand l'identificateur est renvoyé à la suite d'une

[2] Si le système de gestion par fournisseur gère les systèmes BPON 100-200 (exigence de dimensionnement souhaitable) et prend entièrement en charge la base de données de gestion partagées contenue dans les Recommandations UIT-T Q.843.1 et Q.834.2, alors le nombre d'entités gérées prises en charge par le système de gestion par fournisseur varie entre des dizaines de millions et des milliards, selon les offres de service.

opération de gestion (ou compris dans des notifications) l'on part du principe que cet identificateur est "connu" lors d'invocations ultérieures du service de gestion par le système OMS. Dans la présente spécification, l'identificateur d'une entité gérée a reçu une syntaxe qui permet au système de gestion par fournisseur d'informer le système OMS d'au moins un des trois modes d'interaction avec l'entité gérée lors d'invocations ultérieures. Si l'entité gérée est un objet géré disponible à l'interface IF1, alors le nom CORBA de l'objet est fourni comme décrit dans la Rec. UIT-T X.780. Si l'entité gérée dispose d'une façade prenant en charge cet objet à l'interface IF1, alors le nom CORBA de l'objet de façade est fourni comme décrit dans la Rec. UIT-T X.780.1. Si aucune de ces capacités n'est encore disponible, alors l'identificateur se compose d'une référence unique dans le contexte du domaine de gestion du système de gestion par fournisseur. En général, les détails de cette référence sont fondés sur une hiérarchie de confinement spécifique, déterminée par un accord entre fournisseur et opérateur dont les détails sont hors du domaine d'application de la présente Recommandation.

8 Organisation des fichiers en langage IDL

La spécification d'interface proposée dans la présente Recommandation se compose d'un module de rattachement appelé "q834_4". Ce module de rattachement est subdivisé en fichiers de langage IDL (sous-modules) correspondant à des sous-modules nommés. Chacun de ces fichiers contient une ou plusieurs définitions d'interface représentant un service de gestion spécifique qui est pris en charge à l'interface IF1. Le Tableau 1 énumère, en les associant, le numéro de référence selon l'Annexe C, le nom de fichier en langage IDL, les interfaces en langage IDL confinées, et une référence à la description du scénario associé de la Rec. UIT-T Q.834.3.

Tableau 1/Q.834.4 – Organisation du module q834_4

Référence d'Annexe	Fichiers en langage IDL	Interface en langage IDL	Titre du scénario
C.1	Q834AccessControl	AccessControlMgr	Contrôle d'accès
C.2	Q834Build	Builder	Construction de ressources de réseau BPON
C.3	Q834Common	ProbableCause MonitoringParameter RecordSetType PMCategory PhysicalLayerLoopback	
C.4	Q834ControlArchive	RecordSetMgr	Archivage des commandes
C.5	Q834SoftwareDownload	DownloadMgr	Distribution des logiciels
		VersionRepository	Contrôle de la version des logiciels d'élément de réseau
C.6	Q834EventPublisher	AlarmEventSupplier SecurityEventSupplier DiscoveryEventSupplier	Publication d'événements
C.7	Q834MIBTransfer	MIBMover	Restauration d'élément de réseau

Tableau 1/Q.834.4 – Organisation du module q834_4

Référence d'Annexe	Fichiers en langage IDL	Interface en langage IDL	Titre du scénario
C.8	Q834PerformanceManager	ImpairmentPersistence	RCAA & RCIA
		ReportController	Commande de signalisation pour la surveillance de la qualité de fonctionnement et du trafic
C.9	Q834ProfileManager	ProfileConsumer ProfileUsageMgr ProfileRetriever	Gestion des objets de profil
C.10	Q834Registrar	NERegistrar	Fourniture des ressources de réseau BPON installées
C.11	Q834ResourceAllocation	ResourceAllocator	Réservation des ressources
C.12	Q834SchedulerManagement	SchedulerMgr	Programmateur
C.13	Q834ServiceProvisioning	ServiceProvisioner	Fourniture du service
C.14	Q834Synchroniser	Synchroniser	Fourniture des listes récapitulatives d'événements en cours
			Synchronisation d'élément de réseau
C.15	Q834Test	TestActionPerformer	Essais
C.16	Q834FileTransfer	TransferMgr	Transfert en masse

9 Modules

9.1 Module AccessControl

Ce module décrit la fonctionnalité pour la création, la suppression, l'attribution et l'utilisation des informations de contrôle d'accès pour des opérateurs qui utilisent l'application de client d'interface GUI du système de gestion par fournisseur. Dans ce module, des utilisateurs peuvent être attribués à des groupes d'utilisateurs. Des permissions peuvent être accordées à des utilisateurs individuels ou à des groupes d'utilisateurs. L'utilisateur possède les permissions de tout groupe auquel il a été attribué. Toute permission définie individuellement (voir § 9.1.1.14 et 9.1.1.15) pour un utilisateur spécifique a priorité sur une attribution de groupe. Dans les attributions de groupe, le plus haut niveau de permission prévaut pour tout utilisateur attribué aux groupes. L'on part du principe que l'utilisateur privilégié ne demandera pas d'activités cibles ambiguës lors de la création ou de la modification d'un groupe d'utilisateurs ou lors de la spécification de la permission accordée à un utilisateur individuel.

Une activité cible est définie comme le décrit le Tableau 2 dans tout le module:

Tableau 2/Q.834.4 – Détail d'une activité cible

Nom du champ	Définition	Syntaxe	Commentaires
activityLevel	Spécifie le niveau d'accès de l'activité	enum	monitorOnly allowedToExecute noAccess
activityType	Spécifie le type de l'activité	short	Activité définie sous la forme de diverses constantes dans l'interface AccessControlMgr
administrationDomainList	Identificateur fourni par le système OMS ou par l'opérateur pendant l'inscription afin d'indiquer le domaine administratif auquel l'élément de réseau appartient	UserLabel	

9.1.1 Interface AccessControlMgr

9.1.1.1 setPasswordPolicy

Cette opération permet aux utilisateur privilégiés de gérer la politique relative aux mots de passe.

La signature de l'opération **setPasswordPolicy** est indiquée ci-dessous:

```
void setPasswordPolicy (in PasswordPolicyType passwordPolicy)
                        raises (AccessDenied);
```

Le paramètre d'entrée **passwordPolicy** désigne la politique relative aux mots de passe qui est implémentée par le système de gestion du fournisseur. Ce paramètre se compose des éléments UserLoginPolicy et SessionPolicy. L'élément UserLoginPolicy régit les paramètres suivants concernant l'identifiant et le mot de passe de l'utilisateur: la longueur minimale de l'identifiant d'utilisateur, la longueur minimale du mot de passe d'utilisateur, la durée en jours avant qu'un mot de passe puisse être réutilisé, le nombre maximal de tentatives de légitimation autorisées avant que l'accès de l'utilisateur soit bloqué pendant un jour, la durée de validité du mot de passe (en jours), l'indication si le mot de passe doit absolument contenir une association de lettres et de chiffres, si au moins un caractère spécial est requis, si le mot de passe peut contenir une répétition de caractères et si l'identifiant d'utilisateur peut faire partie du mot de passe.

L'élément SessionPolicy désigne les paramètres suivants concernant la session: la durée pendant laquelle une session est inactive avant d'être interrompue, la durée pendant laquelle un identifiant d'utilisateur inactif est désactivé et le nombre maximal de sessions actives autorisées pour chaque identifiant d'utilisateur.

Etant donné qu'une seule politique de mot de passe peut exister dans un système de gestion par fournisseur, il n'est pas nécessaire d'avoir une méthode de création: soit une politique par défaut existe dans le système de gestion par fournisseur ou la première méthode réglée peut remplir la fonction de réglage par création.

La valeur de retour est de type **void**.

9.1.1.2 passwordPolicyGet

Cette opération permet aux utilisateurs privilégiés d'extraire la politique concernant la syntaxe des données échangées et les temporisations utilisées lors de l'entrée en session du système de gestion par fournisseur.

La signature de l'opération **passwordPolicyGet** est indiquée ci-dessous:

```
PasswordPolicyType passwordPolicyGet()
                   raises (AccessDenied);
```

Aucun paramètre d'entrée n'est requis par cette opération.

La valeur de retour est de type **PasswordPolicyType** et indique les détails régissant l'entrée en session de l'utilisateur dans le système de gestion par fournisseur.

9.1.1.3 userListGet

Cette opération permet aux utilisateurs privilégiés d'extraire la liste des identifiants d'utilisateur ayant une certaine forme d'accès au système de gestion par fournisseur ainsi que leurs activités cibles et leurs appartenances à des groupes.

La signature de l'opération **userListGet** est indiquée ci-dessous:

```
UserSeqType userListGet ()
                   raises (AccessDenied);
```

Aucun paramètre d'entrée n'est requis par cette opération.

La valeur de retour est de type **UserSeqType** et indique la liste des identifiants d'utilisateur ayant une certaine forme d'accès au système de gestion par fournisseur ainsi que leurs activités cibles et appartenances à des groupes.

9.1.1.4 userGroupListGet

Cette opération permet aux utilisateurs privilégiés d'extraire les groupes d'utilisateurs ayant accès au système de gestion par fournisseur avec les membres du groupe et les activités cibles identifiés.

La signature de l'opération **userGroupListGet** est indiquée ci-dessous:

```
UserGroupSeqType userGroupListGet ()
                   raises (AccessDenied);
```

Aucun paramètre d'entrée n'est requis par cette opération.

La valeur de retour est de type **UserGroupSeqType** et indique les groupes d'utilisateurs ayant accès au système de gestion par fournisseur avec les membres du groupe et les activités cibles identifiés.

9.1.1.5 userGet

Cette opération permet aux utilisateurs privilégiés d'extraire l'appartenance de l'utilisateur à un groupe et ses activités cibles.

La signature de l'opération **userGet** est indiquée ci-dessous:

```
UserType userGet (
               in UserIdType userId )
               raises (AccessDenied, UnknownUserIds);
```

Le paramètre d'entrée **userId** désigne l'utilisateur considéré.

La valeur de retour est de type **UserType** et indique l'appartenance de l'utilisateur à un groupe et ses activités cibles.

9.1.1.6 userGroupGet

Cette opération permet aux utilisateurs privilégiés d'extraire les utilisateurs membres du groupe et leurs activités cibles autorisées.

La signature de l'opération **userGroupGet** est indiquée ci-dessous:

```
UserGroupType userGroupGet (
                    in UserLabelType userGroupId)
                    raises (AccessDenied, UnknownUserGroupId);
```

Le paramètre d'entrée **userGroupId** désigne le groupe d'utilisateurs.

La valeur de retour est de type **UserGroupType** et indique l'identificateur du groupe d'utilisateurs, les utilisateurs membres du groupe et les activités cibles autorisées.

9.1.1.7 createUserGroup

Cette opération permet aux utilisateurs privilégiés de créer un nouveau groupe d'utilisateurs. Ce groupe d'utilisateurs sert à donner à certains utilisateurs accès à certains éléments de réseau. L'utilisateur privilégié crée le nouveau groupe en donnant la liste des activités permises au groupe d'utilisateurs.

La signature de l'opération **createUserGroup** est indiquée ci-dessous:

```
void createUserGroup (
                    in UserLabelType userGroupId,
                    in TargetActivitySeqType targetAdditions)
                    raises (DuplicateUserGroupId, UnknownTargets, AccessDenied);
```

Le paramètre d'entrée **userGroupId** désigne le groupe à créer. L'identificateur UserGroupId n'est pas autorisé à avoir la valeur de chaîne vide. Le paramètre d'entrée **targetAdditions** désigne les activités cibles permises aux utilisateurs appartenant à ce groupe.

La valeur de retour est de type **void**.

9.1.1.8 modifyUserGroup

Cette opération permet aux utilisateurs privilégiés d'ajouter/de supprimer les activités cibles d'un groupe d'utilisateurs. L'utilisateur privilégié spécifie le groupe d'utilisateurs et la liste des activités cibles à ajouter ou à supprimer. Le système de gestion par fournisseur traite les suppressions avant les adjonctions. Cela permet par exemple de relever le niveau d'autorisation d'une activité donnée.

La signature de l'opération **modifyUserGroup** est indiquée ci-dessous:

```
TargetActivitySeqType modifyUserGroup (
                        in UserLabelType userGroupId,
                        in TargetActivitySeqType targetAdditions,
                        in TargetActivitySeqType targetDeletions)
                        raises (UnknownUserGroupId, UnknownTargets,
                        AccessDenied);
```

Le paramètre d'entrée **userGroupId** désigne le groupe d'utilisateurs à destination/en provenance duquel un opérateur souhaite ajouter/supprimer des activités cibles. Le paramètre d'entrée **targetAdditions** désigne les activités cibles qui ont besoin d'être ajoutées au groupe. Le paramètre d'entrée **targetDeletions** désigne les activités cibles qui ont besoin d'être supprimées du groupe.

La valeur de retour est de type **TargetActivitySeqType** et indique la liste mise à jour des activités cibles qui sont permises aux utilisateurs appartenant à ce groupe d'utilisateurs.

9.1.1.9 deleteUserGroup

Cette opération permet aux utilisateurs privilégiés de supprimer un groupe d'utilisateurs existant. La suppression ne peut être effectuée que si le groupe d'utilisateurs est vide.

La signature de l'opération **deleteUserGroup** est indiquée ci-dessous:

```
void deleteUserGroup (
              in UserLabelType userGroupId)
              raises (AccessDenied, UserGroupNotEmpty, UnknownUserGroupId);
```

Le paramètre d'entrée **userGroupId** désigne le groupe à supprimer.

La valeur de retour est de type **void**.

9.1.1.10 addUsersToGroup

Cette opération permet à l'utilisateur privilégié d'ajouter de nouveaux utilisateurs à un groupe d'utilisateurs existant.

La signature de l'opération **addUsersToGroup** est indiquée ci-dessous:

```
void addUsersToGroup (
              in UserLabelType userGroupId,
              in UserIdSeqType userIdList)
              raises (AccessDenied, UnknownUserGroupId);
```

Le paramètre d'entrée **userGroupId** désigne le groupe auquel de nouveaux utilisateurs ont besoin d'être ajoutés. Le paramètre d'entrée **userIdList** désigne un ensemble d'identificateurs d'utilisateur qui ont besoin d'être ajoutés au groupe d'utilisateurs existant.

La valeur de retour est de type **void**.

9.1.1.11 deleteUsersFromGroup

Cette opération permet aux utilisateurs privilégiés de supprimer des utilisateurs d'un groupe d'utilisateurs.

La signature de l'opération **deleteUsersFromGroup** est indiquée ci-dessous:

```
void deleteUsersFromGroup (
              in UserLabelType userGroupId,
              in UserIdSeqType userIdList)
              raises (AccessDenied, UnknownUserGroupId, UnknownUserIds );
```

Le paramètre d'entrée **userGroupId** désigne le groupe duquel des utilisateurs ont besoin d'être supprimés. Le paramètre d'entrée **userIdList** désigne un ensemble d'identificateurs d'utilisateur à supprimer du groupe d'utilisateurs.

La valeur de retour est de type **void**.

9.1.1.12 getPermissionList

Cette opération permet aux utilisateurs privilégiés d'obtenir la liste des activités autorisées à un utilisateur spécifié.

La signature de l'opération **getPermissionList** est indiquée ci-dessous:

```
TargetActivitySeqType getPermissionList (
                    in UserIdType userId)
                    raises (UnknownUserIds, AccessDenied);
```

Le paramètre d'entrée **userId** désigne l'utilisateur dont l'utilisateur privilégié souhaite obtenir les activités cibles autorisées.

La valeur de retour est de type **TargetActivitySeqType** et indique la liste des activités cibles qui sont permises à un utilisateur spécifié.

9.1.1.13 modifyPermissionList

Cette opération permet aux utilisateurs privilégiés de modifier la liste des activités permises à un utilisateur. L'utilisateur privilégié spécifie l'identifiant de l'utilisateur et la liste des activités qui ont besoin d'être ajoutées ou supprimées. Le système de gestion par fournisseur traite les suppressions avant les adjonctions. Cela permet par exemple le relèvement du niveau d'autorisation d'une activité donnée.

La signature de l'opération **modifyPermissionList** est indiquée ci-dessous:

```
TargetActivitySeqType modifyPermissionList (
                        in UserIdType userId,
                        in TargetActivitySeqType targetAdditions,
                        in TargetActivitySeqType targetDeletions)
                        raises (UnknownUserIds, UnknownTargets, AccessDenied);
```

Le paramètre d'entrée **userId** désigne l'utilisateur dont l'utilisateur privilégié souhaite modifier les activités autorisées. Le paramètre d'entrée **targetAdditions** désigne les activités autorisées qui ont besoin d'être ajoutées à un utilisateur spécifié. Le paramètre d'entrée **targetDeletions** désigne les activités autorisées qui ont besoin d'être supprimées d'un utilisateur spécifié.

Si un utilisateur appartient à de multiples groupes qui ont la même activité, alors cet utilisateur adopte le niveau d'accès qui est le plus haut. Quand à la fois l'utilisateur et le groupe auquel cet utilisateur appartient sont ajoutés au même domaine administratif, alors le niveau d'activité des utilisateurs a priorité sur le niveau d'activité du groupe.

La valeur de retour est de type **TargetActivitySeqType** et indique la liste des activités qui sont autorisées à un utilisateur spécifié après la modification.

9.1.1.14 createUser

Cette opération permet aux utilisateurs privilégiés de créer un nouvel utilisateur. L'utilisateur privilégié donne un nouvel identifiant d'utilisateur, son mot de passe, et la liste des activités permises à l'utilisateur.

La signature de l'opération **createUser** est indiquée ci-dessous:

```
void createUser (
            in UserIdType userId,
            in PasswordType password,
            in TargetActivitySeqType targetAdditions)
            raises (DuplicateUserId, UnknownTargets, AccessDenied,
            UserLoginPolicyViolation);
```

Le paramètre d'entrée **userId** désigne les informations d'identification à associer au nouvel utilisateur. Le paramètre d'entrée **password** désigne le mot de passe à associer au nouvel identifiant d'utilisateur. Le paramètre d'entrée **targetAdditions** désigne les activités cibles permises au nouvel utilisateur.

La valeur de retour est de type **void**.

9.1.1.15 deleteUser

Cette opération permet aux utilisateurs privilégiés de supprimer un utilisateur existant.

La signature de l'opération **deleteUser** est indiquée ci-dessous:

```
void deleteUser (
            in UserIdType userId)
            raises (UnknownUserIds, AccessDenied);
```

Le paramètre d'entrée **userId** désigne l'utilisateur à supprimer.

La valeur de retour est de type **void**.

9.1.1.16 resetPassword

Cette opération permet aux utilisateurs privilégiés de réinitialiser le mot de passe pour un utilisateur. Cette opération intervient quand l'ancien mot de passe n'est pas disponible et que l'utilisateur privilégié a besoin de fixer le nouveau mot de passe pour l'utilisateur. A la première utilisation, le système de gestion par fournisseur invitera l'utilisateur à changer son mot de passe.

La signature de l'opération **resetPassword** est indiquée ci-dessous:

```
void resetPassword (
               in UserIdType userId,
               in PasswordType newPassword)
               raises (UnknownUserIds, UserLoginPolicyViolation, AccessDenied);
```

Le paramètre d'entrée **userId** désigne l'utilisateur qu'il convient de supprimer. Le paramètre d'entrée **newPassword** spécifie le mot de passe qu'il convient d'associer à l'identifiant de l'utilisateur.

La valeur de retour est de type **void**.

9.1.1.17 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **DuplicateUserId** est déclenchée quand l'identifiant de l'utilisateur spécifié pour la nouvelle création d'un utilisateur existe déjà dans la base de données du système de gestion par fournisseur.

L'exception **DuplicateUserGroupId** est déclenchée quand l'identificateur de groupe d'utilisateurs spécifié pour la nouvelle création d'un groupe d'utilisateurs existe déjà dans le système de gestion par fournisseur. En d'autres termes, le système de gestion par fournisseur surveille, en fonction d'une politique, l'attribution d'étiquettes d'utilisateur à des identificateurs de groupe d'utilisateurs pour tous les utilisateurs reconnus par ce système.

L'exception **UnknownTargets** est déclenchée quand la liste spécifiée des activités cibles ne peut pas être identifiée.

L'exception **UnknownUserGroupId** est déclenchée quand l'identificateur du groupe d'utilisateurs spécifié ne peut pas être identifié.

L'exception **UnknownUserIds** est déclenchée quand aucun identificateur d'utilisateur n'est reconnu.

L'exception **UserGroupNotEmpty** est déclenchée quand il existe une tentative de supprimer un groupe d'utilisateurs mais que ce groupe d'utilisateurs n'est pas vide (c'est-à-dire qu'il y a des utilisateurs inscrits dans le groupe).

L'exception **UserLoginPolicyViolation** est déclenchée quand le mot de passe spécifié pour un nouvel utilisateur ou pour le changement de mot de passe viole la politique d'ouverture de session de l'utilisateur.

9.2 Module Build

Le système de gestion par fournisseur construit des groupements de modèles de gestion pour des équipements planifiés à la demande d'un système OMS ou d'un opérateur dans le cadre d'activités de préfourniture. Ces ressources comprennent des nœuds (terminaisons OLT, ONT, ONU, NT) et des modules d'extension. Si l'équipement est installé, des opérations de modification servent à effectuer la fourniture des ressources installées. Des groupements de protection sont établis pour des ressources préfournies ou installées.

9.2.1 Interface avec le constructeur

9.2.1.1 buildNode

Cette opération construit un élément de réseau dans le système de gestion par fournisseur. Un ensemble d'entités gérées est automatiquement créé dans le modèle d'informations de gestion conservé par le système de gestion du fournisseur en conséquence de cette opération. Les instances d'objets de maintenance *equipmentHolderF* pour alvéoles et logements, les instances de réseau NEFSAN et les instances d'extrémité de trajet physique *physicalPathTPF* pour accès intégrés sont des exemples de ce qui peut être automatiquement créé, selon l'implémentation de l'équipement du fournisseur.

Si l'élément de réseau est une terminaison ONT ou ONU, cette activité supprime automatiquement, de la largeur de bande disponible du système, la largeur de bande nécessaire afin de prendre en charge le canal d'exploitation intégré entre les terminaisons OLT et ONT ou ONU.

La signature de l'opération **buildNode** est indiquée ci-dessous:

```
ManagedEntityIdType buildNode (
                    in NEKindType nEKind,
                    in string supplierName,
                    in string location,
                    in VersionType version,
                    in SerialNumType serialNum,
                    in NameSeqType alarmSeverityProfiles,
                    in NameSeqType thresholdDataProfiles,
                    in SlotAssignmentSeqType slotAssignmentList,
                    in ManagedEntityIdType port,
                    in string modelCode,
                    in string systemTitle,
                    in VersionSeqType softwareVersions,
                    in UserLabelType nEUserLabel,
                    in ExternalTimeType externalTime,
                    in SystemTimingType systemTiming,
                    in AdministrationDomainType administrationDomain)
                    raises (UnrecognisedVersion, InvalidSerialNumSyntax,
                    DuplicateSerialNumber, UnknownProfiles,
                    UnknownManagedEntity, DuplicateUserLabel, AccessDenied,
                    InvalidExternalTime, UnknownSystemTimingSource,
                    ProfileSuspended);
```

Le paramètre d'entrée **nEKind désigne** le type d'élément de réseau à construire. Les choix possibles de type d'élément de réseau sont les suivants: OLT, ONT, ONU, ou NT. Les entrées **supplierName** et **version** identifient le vendeur de l'élément de réseau et la version logicielle de l'élément de réseau à construire. Le paramètre d'entrée **location** désigne l'emplacement physique de l'élément de réseau planifié. L'entrée **serialNum** donne une chaîne unique correspondant à l'élément de réseau. L'entrée **alarmSeverityProfiles** désigne les profils servant à configurer la sévérité d'alarme pour des alarmes individuelles qui doivent être signalées par l'élément de réseau. L'entrée **thresholdDataProfiles** désigne les profils à utiliser lors de la configuration de valeurs de seuil afin de produire les alertes TCA de l'élément de réseau. L'entrée **slotAssignmentList** désigne les attributions acceptables de module d'extension pour les logements de l'élément de réseau. Si un numéro de logement n'est pas mentionné dans la liste, alors l'on part du principe qu'aucune attribution n'a été faite pour lui et qu'il n'y a donc aucune restriction quant au type de carte qui peut être placé dans le logement. Il en est de même si la valeur du module d'extension est la chaîne vide. L'accès **port** d'entrée désigne l'accès PON de la terminaison OLT si l'élément de réseau en construction est une terminaison ONT ou ONU. L'accès **port** d'entrée désigne l'accès de la terminaison ONU desservant la terminaison NT si celle-ci est en construction. En cas de construction d'une terminaison OLT, la valeur de l'accès d'entrée est vide. L'entrée **modelCode** donne une chaîne unique identifiant le type de ressource réseau. Ce paramètre d'entrée est principalement considéré lors de la préfourniture d'une

terminaison ONT ou NT. L'entrée **systemTitle** identifie une étiquette précisée par l'opérateur qui doit être appliquée au nœud. L'entrée **softwareVersions** indique les versions de logiciel qui doit être utilisé par le nœud. L'entrée **nEUserLabel** donne une désignation unique d'opérateur pour l'élément de réseau construit. L'entrée **externalTime** règle l'heure actuelle exprimée en temps généralisé pour l'élément de réseau. L'entrée **systemTiming** désigne la source d'horloge d'entrée de l'élément de réseau qui sera utilisée pour la synchronisation du rythme. Chaque fois que le message "valeur non spécifiée" est transmis pour le type "enum" de données **systemTiming** du système OMS, l'élément de réseau doit utiliser sa source de rythme par défaut. L'entrée **administrationDomain** désigne le domaine auquel l'élément de réseau appartient.

La valeur de retour de type **ManagedEntityIdType** donne un identificateur pour le nouvel élément de réseau créé par l'opération.

9.2.1.2 assignUserLabelsToNE

Cette opération attribue des désignations administratives d'opérateur aux éléments de réseau. Cette opération est requise quand le système OMS est informé pour la première fois de l'élément de réseau par autodécouverte.

La signature de l'opération **assignUserLabelsToNE** est indiquée ci-dessous:

```
void assignUserLabelsToNE (
               in SerialNumType serialNum,
               in UserLabelType nEUserLabel,
               in AdministrationDomainType administrationDomain)
               raises (InvalidSerialNumSyntax, DuplicateSerialNumber,
               DuplicateUserLabel, AccessDenied);
```

L'entrée **serialNum** désigne un élément de réseau spécifique. L'entrée **nEUserLabel** fournit à l'élément de réseau un nom attribué par l'opérateur. L'entrée **administrationDomain** désigne le domaine auquel l'élément de réseau est attribué.

La valeur de retour est de type **void**.

9.2.1.3 modifyNode

Cette opération lance la reconfiguration et la mise à jour de paramètres spécifiques associé à un élément de réseau.

La signature de l'opération **modifyNode** est indiquée ci-dessous:

```
void modifyNode (
            in ManagedEntityIdType managedEntityId,
            in SlotAssignmentSeqType newSlotAssignmentList,
            in NameSeqType newAlarmSeverityProfiles,
            in NameSeqType newThresholdDataProfiles,
            in ManagedEntityIdType port,
            in string newModelCode,
            in UserLabelType newNEUserLabel,
            in ExternalTimeType externalTime,
            in AdministrationDomainType administrationDomain)
            raises(UnknownManagedEntity, UnknownNE, InvalidSlotAssignmentList,
            UnknownProfiles, DuplicateUserLabel, AccessDenied,
            InvalidExternalTime, ProfileSuspended );
```

L'entrée **ManagedEntityId** désigne l'élément de réseau cible à modifier. L'élément de réseau **newSlotAssignmentList** désigne une nouvelle attribution de modules d'extension acceptables pour les logements de l'élément de réseau. Si un numéro de logement n'est pas mentionné dans la liste, alors l'on part du principe qu'il n'y a pas de changement pour l'attribution de ce logement. Si la valeur du module d'extension est la chaîne vide alors il n'y a aucune restriction quant au type de carte qui peut être placé dans le logement et, si un module d'extension est ultérieurement retiré du

logement, aucune alarme de retrait n'est déclenchée. L'entrée **alarmSeverityProfiles** désigne les profils servant à configurer la sévérité des alarmes individuelles qui doivent être signalées par l'élément de réseau. L'entrée **thresholdDataProfiles** désigne les profils à utiliser lors de la configuration des valeurs de seuil afin de produire les alertes TCA de l'élément de réseau. L'accès **port** d'entrée désigne l'accès de réseau PON sur la terminaison OLT s'il existe une modification dans la relation entre la terminaison ONT ou ONU planifiée et l'accès de la terminaison OLT. L'accès **port** d'entrée désigne l'accès de terminaison ONU s'il existe une modification dans la relation entre la terminaison NT planifiée et l'accès de terminaison ONU. L'entrée **newModelCode** donne une chaîne unique identifiant le type de ressource réseau. Ce paramètre d'entrée est principalement considéré quand le nœud modifié est une terminaison ONT ou NT. L'entrée **newNEUserLabel** donne une nouvelle étiquette d'utilisateur à appliquer à l'élément de réseau. L'entrée **externalTime** donne un nouveau temps de référence actuel pour l'élément de réseau. L'entrée **administrationDomain** désigne le domaine auquel l'élément de réseau est réattribué.

La valeur de retour est de type **void**.

9.2.1.4 deleteNode

Cette opération supprime l'élément de réseau du système de gestion par fournisseur et annule une demande antérieure de préfourniture. En conséquence de cette opération, toutes les entités gérées, automatiquement créées en conséquence de l'opération buildNode correspondante, sont également supprimées. Cela déclenchera l'exception **RemainingContainedManagedEntities** s'il reste d'autres entités gérées confinées, telles que des connexions de service. Toutes les éventuelles réservations de largeur de bande associées à ce nœud sont également supprimées.

La signature de l'opération **deleteNode** est indiquée ci-dessous:

```
void deleteNode (
            in ManagedEntityIdType managedEntityId)
            raises (UnknownNE, RemainingContainedManagedEntities, AccessDenied,
            RemainingReservations, RemainingSubnetworkConnections);
```

L'entrée **managedEntityId** désigne l'élément de réseau à supprimer par cette opération.

La valeur de retour est de type **void**.

9.2.1.5 modifyPort

Cette opération modifie les paramètres de l'accès désigné, indiqués par l'identificateur **physicalPathTPId**.

La signature de l'opération **modifyPort** est indiquée ci-dessous:

```
void     modifyPort    (
              in ManagedEntityIdType physicalPathTPId,
              in NameSeqType newAlarmSeverityProfiles,
              in NameSeqType newThresholdDataProfiles,
              in NameSeqType newPortProfiles,
              in string newFrameFormat,
              in AdministrativeStateType administrativeState,
              in OpticalWaveLengthArraySeqType newOpticalWavelengthArray,
              in LoopbackLocationIdSeqType newLoopbackLocationIds,
              in unsigned long newInterfaceSpeed,
              in unsigned long aRCTimer)
              raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
              InterfaceSpeedNotChangeable, ProfileSuspended );
```

L'entrée **physicalPathTPId** désigne l'accès à modifier. L'entrée **alarmSeverityProfiles** désigne les profils servant à configurer la sévérité des alarmes individuelles qui doivent être signalées par l'élément de réseau. L'entrée **thresholdDataProfiles** désigne les profils à utiliser lors de la configuration de valeurs de seuil afin de produire les alertes TCA de l'élément de réseau. Le

paramètre d'entrée **newPortProfiles** désigne les profils servant à terminer la fourniture de l'accès. Les exemples de tels profils comprennent mais ne sont pas limités à ce qui suit: ATMNetworkAccessProfile, UNIInfo, EthernetProfile, CESServiceProfile, MACBridgeServiceProfile, LESServiceProfile, AAL 1Profile et AAL 5Profile. L'entrée **newFrameFormat** désigne un nouveau format de trame qui sera reçu et émis à l'accès physique à condition que le format de trame soit configurable. L'entrée **administrativeState** spécifie le réglage modifié pour ce paramètre. L'entrée **newLoopbackLocationIds** désigne de nouveaux identificateurs de localisation pour l'essai de rebouclage associé à l'accès physique. L'entrée **newInterfaceSpeed** désigne une nouvelle vitesse d'interface de l'accès physique à condition que celui-ci soit configurable. Le paramètre d'entrée **aRCTimer** donne le temps non négatif (en secondes) pendant lequel la ressource réseau détecte initialement un signal valide avant de déclencher d'éventuelles alarmes de communication à cet accès.

La valeur de retour est de type **void**.

9.2.1.6 buildPlugInUnit

Cette opération construit un module d'extension dans le système de gestion par fournisseur dans le cadre des activités de préfourniture. Un ensemble d'entités gérées est automatiquement créé dans le modèle d'information de gestion conservé par le système de gestion du fournisseur en conséquence de cette opération. Selon le type de module d'extension, diverses terminaisons sont automatiquement créées. L'attribution de logement du nœud confinant est automatiquement mise à jour afin de comprendre la modification requise par cette opération.

La signature de l'opération **buildPlugInUnit** est indiquée ci-dessous:

```
ManagedEntityIdType buildPlugInUnit (
                        in ManagedEntityIdType nEId,
                        in NameType alarmSeverityProfile,
                        in UserLabelType plugInUnitUserLabel,
                        in string modelCode,
                        in AdministrativeStateType administrativeState,
                        in ManagedEntityIdType equipmentHolder)
                        raises (UnknownNE, DuplicateUserLabel,
                        AccessDenied, UnknownManagedEntity,
                        InvalidEquipmentCode, SlotAlreadyAssigned, UnknownSlot,
                        InvalidSlotAssignmentList, UnknownProfiles,
                        ProfileSuspended );
```

L'entrée **nEId** donne un nom unique pour l'élément de réseau qui contient le module d'extension à construire. L'entrée **plugInUnitUserLabel** donne un identificateur pour le module d'extension à construire. L'entrée **alarmSeverityProfile** désigne le profil servant à configurer les attributions de sévérité d'alarme pour les pannes d'équipement relatives au module d'extension. L'entrée **modelCode** désigne le type du module d'extension. L'entrée **administrativeState** spécifie le réglage initial pour ce paramètre. L'entrée **equipmentHolder** désigne l'emplacement du logement que le module d'extension construit occupera.

La valeur de retour de type **ManagedEntityIdType** donne un unique identificateur pour la carte d'équipement de ligne construite.

9.2.1.7 modifyPlugInUnit

Cette opération modifie les attributs du module d'extension dans le système de gestion par fournisseur.

La signature de l'opération **modifyPlugInUnit** est indiquée ci-dessous:

```
ManagedEntityIdType     modifyPlugInUnit (
                    in ManagedEntityIdType plugInUnitId,
                    in NameType alarmSeverityProfile newAlarmSeverityProfile,
```

```
                    in string newModelCode,
                    in ManagedEntityIdType newEquipmentHolder,
                    in UserLabelType newPlugInUnitUserLabel,
                    in AdministrativeStateType newAdministrativeState)
                    raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
                    InvalidEquipmentCode, SlotAlreadyAssigned, UnknownSlot,
                    InvalidSlotAssignmentList, InvalidUserLabelSyntax,
                    ProfileSuspended);
```

L'entrée **plugInUnitId** désigne l'entité gérée à modifier par cette opération. L'entrée **newAlarmSeverityProfile** modifie le profil servant à configurer les attributions de sévérité d'alarme pour les pannes d'équipement relatives au module d'extension. L'entrée **newModelCode** modifie le type du module d'extension. L'entrée **newEquipmentHolder** modifie l'attribution de logement pour le module d'extension. L'entrée **newPlugInUserLabel** donne une nouvelle étiquette d'utilisateur pour le module d'extension. L'entrée **newAdministrativeState** spécifie le réglage modifié pour ce paramètre.

La valeur de retour de type **ManagedEntityIdType** donne un unique identificateur pour la carte d'équipement de ligne modifiée.

9.2.1.8 deletePlugInUnit

Cette opération supprime un module d'extension du système de gestion par fournisseur. En conséquence de cette opération, toutes les entités gérées automatiquement créées par l'opération correspondante buildPlugInUnit sont également supprimées. Cela déclenchera l'exception **RemainingSubnetworkConnections** si d'éventuelles connexions sont encore présentes. Cette opération sert à supprimer les informations de préfourniture qui ne sont plus recherchées par l'opérateur.

La signature de l'opération **deletePlugInUnit** est indiquée ci-dessous:

```
void deletePlugInUnit (
                    in ManagedEntityIdType plugInUnitId)
                    raises (UnknownManagedEntity, RemainingSubnetworkConnections,
                    AccessDenied, RemainingReservations);
```

L'entrée **plugInUnitId** désigne le module d'extension à supprimer.

La valeur de retour est de type **void**.

9.2.1.9 buildProtectionGrouping

Cette opération construit un groupement de protection dans le système de gestion par fournisseur. Un accès ne peut appartenir à plus d'un seul groupement de protection et doit toujours être préfourni avant cette opération. En dehors du domaine d'application de l'interface IF1, le fournisseur et l'opérateur sont parvenus à une compréhension mutuelle des arrangements de protection valides pris en charge par l'équipement du fournisseur. Si un accès protégé est énuméré dans la liste des unités de protection, il doit également y avoir au moins un accès protecteur. Tous les accès doivent avoir les mêmes caractéristiques de trajet physique.

La signature de l'opération **buildProtectionGrouping** est indiquée ci-dessous:

```
ManagedEntityIdType  buildProtectionGrouping (
                        in ProtectionParameterType protectionParameters,
                        in ProtectionUnitSeqType protectionUnitList)
                        raises (InvalidProtectionScheme, AccessDenied );
```

L'entrée **protectionParameters** donne les caractéristiques du mécanisme de protection. L'entrée **protectionUnitList** désigne les accès protecteurs et protégés sur la ressource réseau.

La valeur de retour de type **ManagedEntityIdType** donne un unique identificateur pour la relation construite entre accès protégés et protecteurs.

9.2.1.10 modifyProtectionParameters

Cette opération modifie le mécanisme de protection pour le groupement de protection désigné. Si la valeur m (= nombre d'accès protecteurs) ou n (= nombre d'accès protégés) diminue, il sera nécessaire que le système OMS se prépare à invoquer l'opération modifyProtectionUnitList afin d'éviter une exception de type: mécanisme de protection non valide. Si la valeur m ou n augmente, il n'est pas nécessaire d'invoquer l'opération modifyProtectionUnitList.

La signature de l'opération **modifyProtectionParameters** est indiquée ci-dessous:

```
void      modifyProtectionParameters (
                      in ManagedEntityIdType protectionGroupingId,
                      in ProtectionParameterType newProtectionParameters)
                      raises (UnknownManagedEntity, InvalidProtectionScheme,
                      AccessDenied);
```

L'entrée **protectionGroupingId** désigne l'entité gérée à modifier par cette opération. L'entrée **newProtectionParameters** permet de remplacer les paramètres de protection existants.

La valeur de retour est de type **void**.

9.2.1.11 modifyProtectionUnitList

Cette opération ajoute ou supprime des éléments de la liste des accès protégés et protecteurs pour le groupement de protection désigné. Une suppression de tous les accès protecteurs ne peut se produire que si tous les accès protégés sont également supprimés. Une adjonction d'accès protégés ou protecteurs ne peut pas dépasser les limites de m ou n spécifiées dans la structure de données des paramètres de protection associés au groupement de protection. Un accès ne peut appartenir à plus d'un seul groupement de protection et doit toujours être préfourni avant cette opération. Tous les accès doivent toujours avoir les mêmes caractéristiques de trajet physique.

La signature de l'opération **modifyProtectionUnitList** est indiquée ci-dessous:

```
void      modifyProtectionUnitList (
                     in ManagedEntityIdType protectionGroupingId,
                     in ProtectionUnitSeqType deltaProtectionUnitList,
                     in boolean addDeleteInd)
                     raises (UnknownManagedEntity, InvalidProtectionScheme,
                     AccessDenied);
```

L'entrée **protectionGroupingId** désigne l'entité gérée à modifier par cette opération. L'entrée **deltaProtectionUnitList** indique les modifications apportées aux accès protégés et protecteurs. Le paramètre d'entrée **addDeleteInd** montre si la modification est une adjonction ou une suppression.

La valeur de retour est de type **void**.

9.2.1.12 deleteProtectionGrouping

Cette opération supprime un groupement de protection d'accès du système de gestion par fournisseur. En conséquence de cette opération, toutes les entités gérées, automatiquement créées par l'opération buildProtectionGrouping correspondante, sont également supprimées.

La signature de l'opération **deleteProtectionGrouping** est indiquée ci-dessous:

```
void deleteProtectionGrouping (
                  in ManagedEntityIdType protectionGroupingId)
                  raises (UnknownManagedEntity, AccessDenied);
```

L'entrée **protectionGroupingId** désigne le groupement de protection à supprimer.

La valeur de retour est de type **void**.

9.2.1.13 buildBridge

Cette opération construit un pont de commande MAC dans le système de gestion par fournisseur. Tous les accès doivent être préfournis avant cette opération. Tous les accès d'interface UNI doivent être des accès de réseau LAN. Cette opération n'est pas requise si tous les accès d'interface UNI avec un réseau LAN, contenus dans un élément de réseau, appartiennent automatiquement au même pont et si les réglages par défaut des paramètres du pont sont recherchés par l'opérateur.

La signature de l'opération **buildBridge** est indiquée ci-dessous:

```
ManagedEntityIdType  buildBridge (
                        in NameType mACBridgeProfile,
                        in ManagedEntityIdType uplinkPort,
                        in ManagedEntityIdSeqType uNIPortList)
                        raises (UnknownProfiles, AccessDenied,
                        UnknownManagedEntity, ProfileSuspended);
```

L'entrée **mACBridgeProfile** indique les caractéristiques du pont conforme à la norme ANSI/IEEE Std 802.1D. L'entrée **uplinkPort** désigne l'accès physique à la terminaison OLT assurant l'interface avec le réseau fédérateur de couche IP. Ce paramètre a la valeur de chaîne vide quand il n'y a pas de tels accès assurant l'interface sur la terminaison OLT. L'entrée **uNIPortList** désigne les accès physiques d'interface UNI associés au pont.

La valeur de retour de type **ManagedEntityIdType** fournit un identificateur unique pour le pont construit.

9.2.1.14 modifyBridgeProfile

Cette opération modifie les caractéristiques de la fonction de dérivation en modifiant le profil du service de pont de commande MAC associé au pont.

La signature de l'opération **modifyBridgeProfile** est indiquée ci-dessous:

```
void modifyBridgeProfile  (
                  in ManagedEntityIdType bridgeId,
                  in NameType newMACBridgeProfile)
                  raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
                  ProfileSuspended);
```

L'entrée **bridgeId** désigne l'entité gérée à modifier par cette opération. L'entrée **newMACBridgeProfile** permet de remplacer le profil de pont existant.

La valeur de retour est de type **void**.

9.2.1.15 modifyBridgePortList

Cette opération ajoute ou supprime de la liste des accès d'interface UNI appartenant au groupement de pont. Une suppression de tous les accès d'interface UNI peut se produire. Un accès d'interface UNI ne peut pas être retiré s'il existe une connexion de service active qui lui est associée. Si chaque accès d'interface UNI avec un réseau LAN doit toujours appartenir au même pont, alors cette opération n'est pas requise.

La signature de l'opération **modifyBridgePortList** est indiquée ci-dessous:

```
void modifyBridgePortList  (
                  in ManagedEntityIdType bridgeId,
                  in ManagedEntityIdSeqType deltaUNIPortList,
                  in boolean addDeleteInd)
                  raises (UnknownManagedEntity, RemainingSubnetworkConnections,
                  AccessDenied);
```

L'entrée **bridgeId** désigne l'entité gérée à modifier par cette opération. L'entrée **deltaUNIPortList** indique les modifications à l'accès d'interface UNI recherché. Le paramètre d'entrée **addDeleteInd** montre si la modification est une adjonction ou une suppression.

La valeur de retour est de type **void**.

9.2.1.16 deleteBridge

Cette opération supprime une fourniture de pont due au système de gestion par fournisseur. En conséquence de cette opération, toutes les entités gérées automatiquement créées par l'opération buildBridge correspondante sont également supprimées. Un pont ne peut pas être supprimé s'il reste des connexions de sous-réseau actives qui lui sont associées.

La signature de l'opération **deleteBridge** est indiquée ci-dessous:

```
void deleteBridge (
            in ManagedEntityIdType bridgeId)
            raises (UnknownManagedEntity, AccessDenied,
            RemainingSubnetworkConnections);
```

L'entrée **bridgeId** désigne le pont à supprimer.

La valeur de retour est de type **void**.

9.2.1.17 buildVPNetworkCTP

Cette opération construit une terminaison CTP de réseau de conduits virtuels dans le système de gestion par fournisseur. L'accès contenant cette terminaison CTP de réseau VP doit toujours être préfourni avant cette construction. Cette opération sert à construire des terminaisons CTP de réseau VP afin de prendre en charge des essais d'installation. Les opérations de construction comme de suppression sont requises[3].

La signature de l'opération **buildVPNetworkCTP** est indiquée ci-dessous:

```
ManagedEntityIdType buildVPNetworkCTP (
                        in ManagedEntityIdType port,
                        in short vPI,
                        in NameType trafficDescriptorProfileName,
                        in ATMOverbookingFactorType overbookingFactor,
                        in UserLabelType userLabel,
                        in SegmentEndpointIndType segmentEndpointInd)
                        raises (UnknownProfiles, AccessDenied,
                        UnknownManagedEntity, ParameterViolation,
                        ProfileSuspended);
```

L'accès **port** d'entrée désigne l'interface ATM dans la ressource réseau. L'entrée **vPI** donne la valeur de l'index. L'entrée **trafficDescriptorProfileName** désigne le profil du descripteur de trafic ATM associé à la terminaison CTP. Le paramètre **overbookingFactor** indique en pourcentage le facteur de surréservation qui peut être utilisé dans les algorithmes de contrôle d'admission d'appel pour tous les circuits PVC qui ont la même valeur d'identificateur VPI à l'accès d'interface ATM. Le paramètre d'entrée **userLabel** donne un nom fourni par l'opérateur pour la terminaison VPNetworkCTP. Le paramètre d'entrée **segmentEndpoint** indique si la terminaison CTP est une extrémité de segment.

La valeur de retour de type **ManagedEntityIdType** fournit un identificateur unique pour la terminaison CTP construite.

---

[3] L'opération de suppression est décrite dans le § 9.2.1.18.

9.2.1.18 deleteVPNetworkCTP

Cette opération supprime du système de gestion par fournisseur une fourniture de terminaison CTP dans un réseau de conduits virtuels. En conséquence de cette opération, toutes les entités gérées, automatiquement créées par l'opération buildVPNetworkCTP correspondante, sont également supprimées.

La signature de l'opération **deleteVPNetworkCTP** est indiquée ci-dessous:

```
void deleteVPNetworkCTP (
                    in ManagedEntityIdType vPNetworkCTP)
                    raises (UnknownManagedEntity, AccessDenied);
```

L'entrée **vPNetworkCTP** désigne le pont à supprimer.

La valeur de retour est de type **void**.

9.2.1.19 createdNodesGet

Cette opération extrait la liste des éléments de réseau qui ont été construits au moyen de l'invocation de cette interface.

La signature de l'opération **createdNodesGet** est indiquée ci-dessous:

```
ManagedEntityIdSeqType createdNodesGet () raises (AccessDenied);
```

Il n'y a pas de paramètre d'entrée.

La valeur de retour est de type **ManagedEntityIdSeqType** et indique les nœuds de la liste qui ont été construits par invocation de Q834:Builder interface.

9.2.1.20 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **DuplicateSerialNumber** est déclenchée s'il existe un autre équipement du même type avec ce numéro de série.

L'exception **DuplicateUserLabel** est déclenchée si l'étiquette d'utilisateur fournie dans la demande a servi à étiqueter un autre élément de réseau ou module d'extension. En d'autres termes, le système de gestion par fournisseur est chargé de surveiller les étiquettes d'utilisateur attribuées aux éléments de réseau et aux modules d'extension dans son domaine de gestion.

L'exception **InterfaceSpeedNotChangeable** est déclenchée si l'accès physique ne peut pas prendre en charge la nouvelle vitesse d'interface ou si la vitesse n'est pas configurable.

L'exception **InvalidEquipmentCode** est déclenchée si le code d'équipement n'est pas conforme à la syntaxe acceptée par l'opérateur et le fournisseur.

L'exception **InvalidExternalTime** est déclenchée si le temps externe spécifié n'est pas valide.

L'exception **InvalidProtectionScheme** est déclenchée si la ressource réseau ne prend pas en charge les paramètres de protection spécifiés dans le contexte de l'énumération des accès ou si les unités de protection sont des accès ayant des caractéristiques de trajet physique dissemblables.

L'exception **InvalidSerialNumSyntax** est déclenchée si la syntaxe du numéro de série fourni viole la syntaxe du fournisseur.

L'exception **InvalidSlotAssignmentList** est déclenchée si le logement d'équipement désigné ne peut pas accepter le type requis de module d'extension.

L'exception **InvalidUserLabelSyntax** est déclenchée si l'étiquette d'utilisateur fournie viole des règles de syntaxe commerciales définies par l'opérateur et implémentées dans le système de gestion par fournisseur.

L'exception **ParameterViolation** est déclenchée quand l'identificateur VPI est hors étendue ou en double.

L'exception **ProfileSuspended** est déclenchée quand le ou les profils nommés dans l'invocation ont été suspendus pour utilisation par le système OMS ou par l'opérateur dans le système de gestion par fournisseur.

L'exception **RemainingContainedManagedEntities** est déclenchée s'il reste des entités gérées confinées dans l'entité gérée à supprimer.

L'exception **RemainingReservations** est déclenchée s'il reste des réservations de ressource associées à l'entité gérée à supprimer.

L'exception **RemainingSubnetworkConnections** est déclenchée si l'entité gérée qui est en cours de suppression contient une ou plusieurs connexions de sous-réseau valides.

L'exception **SlotAlreadyAssigned** est déclenchée si le mini-logement requis est déjà préfourni.

L'exception **UnknownManagedEntity** est déclenchée si l'entité gérée désignée est inconnue du système de gestion par fournisseur.

L'exception **UnknownNE** est déclenchée si l'élément de réseau désigné est inconnu du système de gestion par fournisseur.

L'exception **UnknownProfiles** est déclenchée si le nom de profil fourni est inconnu du système de gestion par fournisseur et ne peut pas être extrait du répertoire d'objets de profil.

L'exception **UnknownSlot** est déclenchée si le logement requis est inconnu de l'élément de réseau.

L'exception **UnknownSystemTimingSource** est déclenchée si la source de temps externe est inconnue du système de gestion par fournisseur ou de l'élément de réseau.

L'exception **UnrecognizedVersion** est déclenchée si la version d'équipement indiquée ne concorde pas avec des valeurs connues.

9.3 Module Q834Common

Ce module contient les définitions des types de données utilisés par plus d'un seul des autres modules de langage IDL dans la présente Recommandation. Il indique également trois groupements de constantes. Un grand nombre des types de données mentionnés dans le module Q834Common sont importés de la Rec. UIT-T X.780. Le plus important aspect de ce fichier en langage IDL est la définition du type ManagedEntityIdType. La définition est extraite du fichier en langage IDL et présentée ci-dessous aux fins de la discussion.

```
struct NamingComponentType {
     string type;  // managed entity type
     string id;
};

typedef sequence<NamingComponentType> RDNType;
typedef sequence<RDNType> RDNSeqType;
typedef sequence<NameType>  NameSeqType;

enum IdType {
     none,
     x780_fineGrained,
     x780_coarseGrained
};
```

```
typedef RDNType MEIdType;

struct ManagedEntityIdType {
     IdType id;
     MEIdType mEId;
};
```

La structure ManagedEntityIdType est une valeur de retour pour un grand nombre des opérations dans l'ensemble de la présente Recommandation. L'examen de la définition de ce type de données (en lecture de bas en haut) montre que le système de gestion par fournisseur renvoie une valeur qui fournit une référence soit à un objet géré à granulométrie fine, ou à une façade à granulométrie grossière, ou un identificateur de nom RDN pour une structure de données dans le système de gestion par fournisseur. Dans chacun de ces trois cas, la syntaxe fondamentale de l'identificateur mEId est une séquence de paires de composants de nommage: c'est la syntaxe employée par la Rec. UIT-T X.780 pour les objets gérés et la syntaxe employée par la Rec. UIT-T X.780.1 pour les façades.

9.3.1 Module ProbableCauseConst et interface ProbableCause

Cette définition de module suit le modèle spécifié dans la Rec. UIT-T X.780 et utilise la même syntaxe pour les valeurs de cause probable. Elle indique quelques valeurs propres à la technique employée en plus des valeurs définies dans la Rec. UIT-T X.780.

9.3.2 Interface MonitoringParameter

Cette interface fournit les noms des paramètres de contrôle de la qualité du fonctionnement et du trafic à utiliser par les interfaces AlarmEventSupplier, ImpairmentPersistence et ProfileConsumer. Les valeurs de l'interface MonitoringParameter sont exprimées sous forme de chaînes et font partie des données filtrables contenues dans un événement structuré.

9.3.3 Interface RecordSetType

Cette interface donne la ou les valeurs des données pour la valeur de type *recordset* (chaîne d'articles) des données à utiliser par les interfaces ReportController et RecordSetMgr. Les valeurs de type *recordset* sont exprimées par des valeurs brèves non signées. Les valeurs 1 à 99 sont réservées pour le type HistoryDataType.

9.3.4 Interface PhysicalLayerLoopback

Cette interface donne la ou les valeurs des données pour la valeur de type *loopbackTestType* (type d'essai de rebouclage) des données à utiliser par l'interface TestActionPerformer. Les valeurs de type *recordset* sont exprimées par des valeurs brèves non signées.

9.4 Module ControlArchive

Le système de gestion par fournisseur fournit la fonctionnalité de gestion des journaux pour des groupes spécifiques d'événements y compris l'effacement du contenu du journal. L'utilisateur privilégié peut créer, initialiser, suspendre, reprendre et supprimer des journaux d'événements. Le système de gestion par fournisseur offre également la fonction de commande de l'archivage à court terme des comptes rendus de surveillance de la qualité du fonctionnement et du trafic, y compris l'effacement du contenu de ces chaînes d'articles. Cette fonction comprend également la signalisation de comptes rendus du statut pour les journaux ou chaînes d'articles statistiques en cours. La majorité des archives à court terme contenues dans le système de gestion par fournisseur sont créées automatiquement quand le système de gestion par fournisseur est instancié pour la première fois. Les noms de ces archives, désignés par la syntaxe de type ManagedEntityIdType, peuvent être fournis par le fournisseur à l'opérateur ou être obtenus par invocation de l'opération recordSetListGet avec la valeur "initialList" du type creationModeType et des invocations récurrentes de l'opération getStatusAttributes pour chaque chaîne d'articles désignée dans la première opération.

9.4.1 Interface RecordSetMgr

9.4.1.1 createLog

L'opération createLog sert à créer une chaîne d'articles dans le système de gestion par fournisseur aux fins des informations d'événements d'archivage.

La signature de l'opération **createLog** est la suivante:

```
ManagedEntityIdType createLog (
               in UserLabelType recordSetUserLabel,
               in AdministrativeStateType administrativeState,
               in NameType filterName,
               in FullActionType fullAction,
               in MaxSizeType maxSize,
               in SizeThresholdType sizeThreshold)
               raises (RecordSetExists, DuplicateUserLabel, AccessDenied);
```

Le paramètre d'entrée **recordSetUserLabel** désigne de façon unique la chaîne d'articles contenue dans le système de gestion par fournisseur. Le paramètre d'entrée **administrativeState** spécifie si le nouveau journal est initialisé pour l'enregistrement d'informations d'évènement. Le paramètre d'entrée **filterName** explique les critères d'entrée déterminant quelles notifications d'événement sont enregistrées dans le journal à créer. Si le paramètre **filterName** n'est pas reconnu par le système de gestion du fournisseur, ce paramètre explore les références IOR à la recherche de l'objet de filtrage au moyen du paramètre **filterName** et en consultant le répertoire du service de nommage. Au moyen de la référence IOR, le système est alors en mesure d'extraire les critères d'entrée détaillés de l'interface CosNotifyFilter disponible dans le service de notification. Le paramètre d'entrée **fullAction** spécifie le comportement de la chaîne d'articles quand elle atteint sa longueur maximale. Le paramètre d'entrée **maxSize** spécifie la longueur maximale de la chaîne d'articles. Le paramètre d'entrée **sizeThreshold** spécifie le seuil de longueur de la chaîne d'articles qui déclenche la production d'une alarme ou d'un événement par le système de gestion du fournisseur.

La valeur de retour est de type **ManagedEntityIdType** et indique l'identificateur pour l'archive de journalisation créée par cette opération.

9.4.1.2 createArchive

L'opération createArchive sert à créer une chaîne d'articles pour mémoriser des données chronologiques. L'archivage arrêtera automatiquement son enregistrement quand il atteindra la longueur maximale.

La signature de l'opération **createArchive** est la suivante:

```
ManagedEntityIdType createArchive (
              in UserLabelType recordSetUserLabel,
              in AdministrativeStateType administrativeState,
              in RecordKindType recordKind,
              in MaxSizeType maxSize)
              raises (RecordSetExists, DuplicateUserLabel, AccessDenied);
```

Le paramètre d'entrée **recordSetUserLabel** désigne de façon unique la chaîne d'articles dans le système de gestion par fournisseur. Le paramètre d'entrée **administrativeState** spécifie si la nouvelle archive est initialisée pour l'enregistrement des articles appropriés. Le paramètre d'entrée **recordKind** désigne le type d'article à enregistrer dans la chaîne d'articles. Le paramètre d'entrée **maxSize** spécifie la longueur maximale de la chaîne d'articles.

La valeur de retour est de type **ManagedEntityIdType** et indique l'identificateur pour l'archive créée par cette opération.

9.4.1.3 getStatusAttributes

A tout moment, l'opérateur ou le système OMS peut voir l'état actuel d'une archive à court terme.

La signature de **getStatusAttributes** est la suivante:

```
RecordSetStatusType getStatusAttributes (
                    in ManagedEntityIdType recordSetId)
                    raises (AccessDenied, UnknownRecordSet);
```

Le paramètre d'entrée **recordSetId** désigne de façon unique la chaîne d'articles dans le système de gestion par fournisseur.

La valeur de retour est de type **RecordSetStatusType** qui contient l'état actuel de la chaîne d'articles.

9.4.1.4 suspendArchive

Une fois que l'archive à court terme a été créée et initialisée en vue de son utilisation, le système OMS peut suspendre son utilisation.

La signature de l'opération **suspendArchive** est la suivante:

```
void suspendArchive (
                   in ManagedEntityIdType recordSetId)
                   raises (AccessDenied, UnknownRecordSet);
```

Le paramètre d'entrée **recordSetId** désigne de façon unique la chaîne d'articles dans le système de gestion par fournisseur.

La valeur de retour est de type **void**.

9.4.1.5 resumedArchive

Cette opération reprend l'enregistrement dans une archive ou initialise l'enregistrement dans une chaîne d'articles qui a été construite dans un état de blocage.

La signature de l'opération **resumeArchive** est la suivante:

```
void resumeArchive (
                   in ManagedEntityIdType recordSetId)
                   raises (UnknownRecordSet, AccessDenied);
```

Le paramètre d'entrée **recordSetId** identifie de manière unique la chaîne d'articles dans le système de gestion par fournisseur.

La valeur de retour est de type **void**.

9.4.1.6 deleteArchive

Cette opération supprime une archive du système de gestion par fournisseur.

La signature de l'opération **deleteArchive** est la suivante:

```
void deleteArchive (
            in ManagedEntityIdType recordSetId )
            raises (UnknownRecordSet, AccessDenied );
```

Le paramètre d'entrée **recordSetId** désigne le nom que l'utilisateur privilégié souhaite utiliser afin d'identifier le journal à créer lors d'interactions ultérieures.

La valeur de retour est de type **void**.

9.4.1.7 purgeArchive

Cette opération supprime les informations contenues dans une archive spécifiée. Cependant, l'archive continue son enregistrement.

La signature de l'opération **purgeArchive** est la suivante:

```
void purgeArchive (
              in ManagedEntityIdType recordSetId )
              raises (UnknownRecordSet, AccessDenied );
```

Le paramètre d'entrée **recordSetId** désigne le nom que l'utilisateur privilégié souhaite utiliser afin d'identifier le journal à créer lors d'interactions ultérieures.

La valeur de retour est de type **void**.

9.4.1.8 selectRecords

Après la création d'une archive, le système OMS peut sélectionner certains de ses articles enregistrés.

La signature de l'opération **selectRecords** est la suivante:

```
RecordSeqType selectRecords (
                  in FilterType selectionFilter,
                  in ManagedEntityIdType recordSetId)
                  raises (UnknownRecordSet, Timeout, NoSuchRecords,
                  AccessDenied, TooManyRecords);
```

Le paramètre d'entrée **recordSetId** désigne le nom que l'utilisateur privilégié souhaite utiliser afin d'identifier le journal à créer lors d'interactions ultérieures. Le paramètre d'entrée **selectionFilter** est défini par une chaîne d'articles spécifique comme défini dans la structure de données.

La valeur de retour est de type **RecordSeqType** fournissant les informations requises.

9.4.1.9 recordSetListGet

Cette opération permet à un système OMS d'obtenir une énumération complète des chaînes d'articles gérées par le système de gestion du fournisseur.

La signature de l'opération **recordSetListGet** est la suivante:

```
ManagedEntityIdSeqType recordSetListGet (
                        in CreationModeType creationMode)
                        raises (AccessDenied);
```

Le paramètre d'entrée **creationMode** désigne la façon dont l'archive a été créée, c'est-à-dire si elle a été créée par un opérateur ou l'a été automatiquement dans le cadre de l'initialisation du système de gestion par fournisseur.

La valeur de retour est de type **ManagedEntityIdSeqType** avec énumération des noms des archives à court terme du système de gestion par fournisseur.

9.4.1.10 changeUserLabel

Après la création d'une archive à court terme, le système OMS peut changer l'étiquette d'utilisateur attribuée à l'archive.

La signature de l'opération **changeUserLabel** est la suivante:

```
void changeUserLabel(
            in ManagedEntityIdType recordSetId,
            in UserLabelType newUserLabel)
            raises (UnknownRecordSet, AccessDenied, DuplicateUserLabel);
```

Le paramètre d'entrée **recordSetId** désigne l'archive. Le paramètre d'entrée **newUserLabel** désigne le nouveau nom d'une archive spécifique.

La valeur de retour est de type **void**.

9.4.1.11 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **DuplicateUserLabel** est déclenchée si l'étiquette d'utilisateur fournie dans la demande a servi à étiqueter une autre archive. En d'autres termes, le système de gestion par fournisseur est chargé de surveiller les étiquettes d'utilisateur attribuées à des chaînes d'articles dans son domaine de gestion.

L'exception **LockedAlready** est déclenchée si la valeur réglée de l'état administratif est "locked".

L'exception **NoSuchRecords** est déclenchée si aucun article parmi les chaînes d'articles désignées ne correspond aux critères de sélection.

L'exception **RecordSetExists** est déclenchée si l'archive définie par les paramètres de demande de création existe déjà dans le système de gestion par fournisseur.

L'exception **Timeout** est déclenchée si la durée de traitement a atteint une limite de temporisation définie par le système avant que le processus puisse s'achever.

L'exception **TooManyRecords** est déclenchée si le nombre d'articles sélectionné pour extraction produit une réponse à la demande qui dépasse une longueur prédéterminée.[4]

L'exception **UnknownRecordSet** est déclenchée si la chaîne d'articles désignée est inconnue du système de gestion par fournisseur.

9.5 Module SoftwareDownload

Le processus de téléchargement de logiciel se compose de quatre phases: acheminement, distribution, installation (validation) et activation. Le système de gestion par fournisseur prend en charge ces phases pour le téléchargement de programmes logiciels génériques, de mises à jour logicielles et de modifications logicielles de maintenance (correctifs) apportées aux éléments de réseau dans ce module. Le système de gestion par fournisseur peut accepter des demandes mettant en jeu un ou plusieurs éléments de réseau à la fois. Toutes les activités de gestion logicielle peuvent être individuellement planifiées. Toute demande de téléchargement de logiciels génériques est accompagnée de justificatifs de sûreté par lesquels le système de gestion par fournisseur est autorisé à communiquer avec le serveur de départ. La question de savoir quels justificatifs de sûreté sont utilisés dépend de la réalisation: si la charge logicielle réside dans le système EMS et si ce système la communique (par transfert de fichiers) à l'élément de réseau, alors les justificatifs concernent l'élément de réseau. Si la charge logicielle réside dans le système EMS et si l'élément de réseau la communique, alors les justificatifs concernent le système EMS. Si la charge logicielle réside dans une répertoire distinct, alors les justificatifs de sûreté concernent ce répertoire distinct et l'élément de réseau communique la charge du répertoire.

Les demandes d'activité de gestion de logiciels peuvent provoquer la création d'un objet de suivi de gestion logicielle. La durée de vie de cet objet doit toujours se prolonger jusqu'à ce que toutes les activités associées aient été effectuées avec ou sans succès. L'objet n'est supprimé qu'après une période de rétention prédéfinie et la durée de la période de rétention fait l'objet d'un accord qui est en dehors du domaine d'application de cette interface. La période de rétention relative à l'objet de suivi devrait être assez largement dimensionnée pour comprendre toute activité de récupération

---

4 Cette dimension doit être convenue d'avance par le fournisseur et par l'opérateur.

recherchée. Si l'objet de suivi a été automatiquement supprimé par le système de gestion du fournisseur, alors l'activité de récupération est considérée comme une activité de téléchargement d'une précédente charge logicielle. Le système OMS peut également supprimer cet objet de suivi avant sa terminaison automatique si cela est recherché.

Le système de gestion par fournisseur crée un journal d'exploitation pour chaque activité de distribution logicielle y compris l'acheminement, la validation et l'activation, quel que soit le résultat de l'activité. L'article journalisé contient des résultats détaillés pour chaque cible visée y compris les instants de début et de fin et l'indication de succès ou d'échec.

Le système de gestion par fournisseur conserve une énumération actualisée de toutes les activités logicielles en cours.

Ce module comprend également la prise en charge de l'extraction d'informations sur les versions matérielles et logicielles des éléments de réseau installés. Il est également possible de vérifier qu'un ensemble logiciel peut être téléchargé vers un élément de réseau ou vers un module d'extension spécifique.

9.5.1 Interface DownloadMgr

9.5.1.1 deliverDistSWGlobal

Cette opération demande au système de gestion par fournisseur de télécharger des logiciels génériques à partir d'une source automatique de logiciels aux fins de mises à jour et de modifications logicielles de maintenance (correctifs) apportées aux éléments de réseau. Le système de gestion par fournisseur peut accepter des demandes pour un ou plusieurs éléments de réseau à la fois. L'opération est effectuée au mieux, c'est-à-dire que le système de gestion par fournisseur essaye d'exécuter l'acheminement et la distribution à toutes les cibles spécifiées. Selon le cas, il continue à essayer d'acheminer et de distribuer des logiciels à toutes les cibles nommées même quand il rencontre un échec dans l'un ou l'autre de ces processus pour une cible spécifique. Le journal d'achèvement d'une activité sert à enregistrer les succès et les échecs des tentatives d'acheminement comme de distribution. L'objet SoftwareDownloadTrackingObject continue à être disponible pour des validations et des tentatives d'activation: ces activités ne sont tentées que pour les cibles dans lesquelles l'acheminement et la distribution ont été réussies. L'objet SoftwareDownloadTrackingObject est conservé au-delà de la phase d'activation réussie (en partie afin de permettre des activités de récupération si nécessaire).

Ultérieurement, si une nouvelle terminaison ONT est télémesurée, ou si une rupture de fibre de connexion est réparée, ou si un nouveau module d'extension est installé, et si dans chacun de ces cas la charge logicielle active diffère de celle qui est spécifiée dans cette opération, alors le système de gestion par fournisseur (ou la ressource réseau) sera chargé(e) de mettre à jour les logiciels automatiquement.

La signature de l'opération **deliverDistSWGlobal** est indiquée ci-dessous:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWGlobal (
                        in FilenameSeqType softwareSet,
                        in DCNAddressType softwareSourceAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in ManagedEntityIdSeqType deliverDistTargets)
                        raises (CommFailure, UnrecognisedTarget,
                        InsufficientMemory, SoftwareLoadHWMismatch,
                        SourceUnreachable, UnknownSoftwareLoad, Timeout,
                        AccessDenied, DeniedAccess);
```

Le paramètre d'entrée **softwareSet** désigne la charge logicielle et où celle-ci est située (noms de fichier et chemin d'accès complet) à télécharger. Le paramètre d'entrée **softwareSourceAddr** donne l'adresse du réseau de communication de données (RCD) du serveur où l'ensemble des logiciels

réside. Les paramètres d'entrée **userId** et **password** fournissent le mécanisme d'ouverture de session à la source de logiciels (en supposant que de tels justificatifs de sûreté sont requis). Le paramètre d'entrée **deliverDistTargets** indique la liste des terminaisons OLT où les logiciels doivent être acheminés.

La valeur de retour de type **SoftwareDownloadTrackingObjectIdType** fournit une référence à utiliser lors d'une tentative de validation, d'activation ou de vérification de l'état du processus d'acheminement et de distribution. Le résultat de téléchargement de logiciels est journalisé par le système de gestion du fournisseur.

9.5.1.2 deliverDistSWSpecific

Cette opération est la même que l'opération deliverDistSWGlobal, sauf que le domaine d'application est dans un seul élément de réseau/module d'extension/logement, comme décrit dans le paramètre d'entrée distributionTarget.

La signature de l'opération **deliverDistSWSpecific** est indiquée ci-dessous:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWSpecific (
                        in FilenameSeqType softwareSet,
                        in DCNAddressType softwareSourceAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in TargetType deliverDistTarget)
                        raises (CommFailure, UnrecognisedTarget,
                        InsufficientMemory, SoftwareLoadHWMismatch,
                        SourceUnreachable, UnknownSoftwareLoad, Timeout,
                        AccessDenied, DeniedAccess);
```

Le paramètre d'entrée **softwareSet** désigne la charge logicielle et l'endroit où elle est située (noms de fichier et chemin d'accès complet) à télécharger. Le paramètre d'entrée **softwareSourceAddr** donne l'adresse du réseau de communication de données (RCD) du serveur où réside l'ensemble des logiciels. Les paramètres d'entrée **userId** et **password** fournissent le mécanisme d'ouverture de session à la source de logiciels (en supposant que de tels justificatifs de sûreté sont requis). Le paramètre d'entrée **deliverDistTarget** indique la cible spécifique au niveau de l'élément de réseau/du module d'extension/du logement. Les composants correspondant à la valeur de ce paramètre sont décrits dans le Tableau 3.

Tableau 3/Q.834.4 – Détails du paramètre deliverDistTarget

Nom du champ	Description
containingSystem	Le système est désigné par l'identificateur d'entité gérée de la terminaison OLT en tête de réseau.
containingNE	Désigne la ressource réseau de destination. Quand la valeur est la séquence vide, l'opération distribue les logiciels à tous les éléments de réseau contenus dans le système.
plugInUnitType	Désigne le type de module d'extension. Quand cette valeur est la chaîne vide, la cible de distribution est tous les types appropriés de module d'extension contenus dans l'élément de réseau englobant.
slot	Désigne le logement. Quand cette valeur est la séquence vide, alors tout logement approprié est pris en considération.

La valeur de retour de type **SoftwareDownloadTrackingObjectIdTypeId** donne une référence clé de corrélation à utiliser lors d'une tentative de validation, d'activation ou de vérification de l'état du

processus d'acheminement et de distribution. Le résultat de téléchargement de logiciel est journalisé par le système de gestion du fournisseur.

9.5.1.3 deleteSoftwareDownloadTrackingObject

Cette opération permet au système OMS de signaler au système de gestion par fournisseur que l'objet de suivi logiciel spécifié n'est plus requis.

La signature d'opération **deleteSoftwareDownloadTrackingObject** est indiquée ci-dessous:

```
void deleteSoftwareDownloadTrackingObject  (
            in SoftwareDownloadTrackingObjectIdType   id)
            raises (UnknownSoftwareDownloadTrackingObject, AccessDenied,
            SoftwareTrackingObjectInUse);
```

Le paramètre d'entrée **id** désigne l'objet de suivi logiciel.

La valeur de retour est de type **void**.

9.5.1.4 commit

Cette opération demande au système de gestion par fournisseur d'installer (valider) le logiciel téléchargé à des emplacements désignés.

La signature de l'opération **commit** est indiquée ci-dessous:

```
void commit (
      in SoftwareDownloadTrackingObjectIdType id,
      in TargetType commitTarget)
      raises (InstallationFailure, UnknownSoftwareDownloadTrackingObject,
      AccessDenied, UnrecognisedTarget);
```

Le paramètre d'entrée **id** fournit une référence à l'objet de suivi du téléchargement de logiciels. Le paramètre d'entrée **commitTarget** spécifie la cible spécifique au niveau de l'élément de réseau/du module d'extension/du logement. La cible de validation peut être un sous-ensemble de la cible originale.

La valeur de retour est de type **void**.

9.5.1.5 activate

Cette opération active le logiciel installé à des emplacements désignés.

La signature de l'opération **activate** est indiquée ci-dessous:

```
void activate (
            in SoftwareDownloadTrackingObjectIdType id,
            in TargetType activateTarget)
            raises (UnknownSoftwareDownloadTrackingObject,
            SoftwareNotYetInstalled, ActivationFailure, AccessDenied,
            UnrecognisedTarget);
```

Le paramètre d'entrée **id** fournit une référence à l'objet de suivi du téléchargement de logiciels. Le paramètre d'entrée **activateTarget** spécifie la cible spécifique au niveau de l'élément de réseau/du module d'extension/du logement. La cible d'activation peut être un sous-ensemble de la cible originale.

La valeur de retour est de type **void**.

9.5.1.6 revert

Cette opération active d'anciens logiciels installés à des emplacements désignés. L'opération de récupération n'est logiquement invoquée qu'après activation d'une nouvelle charge logicielle. L'identificateur SoftwareDownloadTrackingObjectId se rapporte au plus récent téléchargement de

logiciel affectant la cible nommée. Si l'opération de récupération est effectuée avec succès, alors la version active devient la version de secours.

La signature de l'opération **revert** est indiquée ci-dessous:

```
void revert (
            in SoftwareDownloadTrackingObjectIdType id,
            in TargetType revertTarget)
            raises (UnknownSoftwareDownloadTrackingObject,
            SoftwareNotYetInstalled, ActivationFailure, AccessDenied,
            UnrecognisedTarget, InvalidSoftwareTrackingObject);
```

Le paramètre d'entrée **id** fournit une référence à l'objet de suivi du téléchargement de logiciels. Le paramètre d'entrée **revertTarget** spécifie la cible spécifique au niveau de l'élément de réseau/du module d'extension/du logement. La cible de récupération peut être un sous-ensemble de la cible originale.

La valeur de retour est de type **void**.

9.5.1.7 getStatus

Cette opération demande le statut des activités logicielles.

La signature de l'opération **getStatus** est indiquée ci-dessous:

```
DownloadStatusSeqType getStatus (
                        in SoftwareDownloadTrackingObjectIdType id)
                        raises (UnknownSoftwareDownloadTrackingObject,
                        AccessDenied);
```

Le paramètre d'entrée **id** fournit une référence à l'objet de suivi du téléchargement de logiciels.

La valeur de retour de type **DownloadStatusSeqType** donne l'état d'avancement de toutes les activités de téléchargement suivies par l'objet de suivi du téléchargement de logiciels.

9.5.1.8 scheduleDeliverDist

Cette opération planifie l'acheminement et la distribution des logiciels à des cibles spécifiées. La distribution des logiciels dans les cibles spécifiées relève de l'implémentation du fournisseur.

La signature de l'opération **scheduleDeliverDist** est indiquée ci-dessous:

```
SoftwareDownloadTrackingObjectIdType scheduleDeliverDist (
                        in FilenameSeqType softwareSet,
                        in DCNAddressType softwareSourceAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in ManagedEntityIdSeqType deliverDistTargets,
                        in GeneralizedTimeType deliverDistStartTime)
                        raises (SoftwareLoadHWMismatch, AccessDenied,
                        InvalidStartTime );
```

Le paramètre d'entrée **softwareSet** désigne la charge logicielle et son emplacement (noms de fichier et chemin d'accès complet) à télécharger. Le paramètre d'entrée **softwareSourceAddr** donne l'adresse du réseau de communication de données (RCD) du serveur où réside l'ensemble des logiciels. Les paramètres d'entrée **userId** et **password** fournissent le mécanisme d'ouverture de session à la source de logiciels (en supposant que de tels justificatifs de sûreté sont requis). Le paramètre d'entrée **deliverDistTargets** indique la liste des terminaisons OLT où les logiciels doivent être acheminés. Le paramètre d'entrée **deliverDistStartTime** fournit l'instant de début planifié.

La valeur de retour de type **SoftwareDownloadTrackingObjectIdType** fournit une référence au processus planifié qui peut servir à suivre l'état d'avancement du processus ou à l'annuler.

9.5.1.9 scheduleCommit

Cette opération planifie l'installation (validation) des logiciels à des cibles prédéterminées.

La signature de l'opération **scheduleCommit** est indiquée ci-dessous:

```
void scheduleCommit (
                  in SoftwareDownloadTrackingObjectIdType
                  deliverDistSoftwareDownloadTrackingObjectId,
                  in GeneralizedTimeType commitStartTime)
                  raises (UnknownSoftwareDownloadTrackingObject,
                  SoftwareNotYetInstalled, AccessDenied, InvalidStartTime );
```

Le paramètre d'entrée **deliverDistSoftwareDownloadTrackingObjectId** fournit une référence à l'objet de suivi de l'acheminement de logiciels. Le paramètre d'entrée **commitStartTime** donne l'instant de début de cette activité planifiée.

La valeur de retour est de type **void.**

9.5.1.10 scheduleActivate

Cette opération planifie l'activation des logiciels à des cibles prédéterminées.

La signature de l'opération **scheduleActivate** est indiquée ci-dessous:

```
void scheduleActivate (
                  in SoftwareDownloadTrackingObjectIdType id,
                  in GeneralizedTimeType activateStartTime)
                  raises (UnknownSoftwareDownloadTrackingObject,
                  SoftwareNotYetInstalled, AccessDenied, InvalidStartTime );
```

Le paramètre d'entrée **id** fournit une référence à l'objet de suivi de l'acheminement de logiciels. Le paramètre d'entrée **activateStartTime** donne l'instant de début de cette activité.

La valeur de retour est de type **void**.

9.5.1.11 cancelScheduledSoftwareActivity

Cette opération annule toutes les activités de téléchargement de logiciels planifiées ultérieurement en association avec cet objet de suivi.

La signature de l'opération **cancelScheduledSoftwareActivity** est indiquée ci-dessous:

```
void cancelScheduledSoftwareActivity (
                        in SoftwareDownloadTrackingObjectIdType id)
                        raises (UnknownSoftwareDownloadTrackingObject,
                        ActivityCompleted, ActivityInProgress, AccessDenied);
```

Le paramètre d'entrée **id** fournit une référence à l'objet de suivi d'activité logicielle.

La valeur de retour est de type **void**.

9.5.1.12 scheduledSoftwareDownloadTrackingObjectListGet

Cette opération extrait la liste des activités programmées en instance de téléchargement de logiciel.

La signature de l'opération **scheduledSoftwareDownloadTrackingObjectListGet** est indiquée ci-dessous:

```
SoftwareDownloadTrackingObjectIdSeqType
scheduledSoftwareDownloadTrackingObjectListGet () raises (AccessDenied);
```

Il n'y a pas de paramètre d'entrée.

La valeur de retour est de type **SoftwareDownloadTrackingObjectIdSeqType** qui donne la liste des activités programmées en instance.

9.5.1.13 onDemandSoftwareDownloadTrackingObjectListGet

Cette opération extrait la liste des activités non programmées de téléchargement de logiciel qui sont en instance.

La signature de l'opération **onDemandSoftwareDownloadTrackingObjectListGet** est indiquée ci-dessous:

```
SoftwareDownloadTrackingObjectIdSeqType
onDemandSoftwareDownloadTrackingObjectListGet () raises (AccessDenied);
```

Il n'y a pas de paramètre d'entrée.

La valeur de retour est de type **SoftwareDownloadTrackingObjectIdSeqType** qui donne la liste des activités non programmées en instance.

9.5.1.14 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **ActivityCompleted** est déclenchée quand l'activité logicielle a été exécutée et ne peut pas être annulée.

L'exception **ActivationFailure** est déclenchée si le processus d'activation des logiciels a échoué même si les logiciels ont été installés (c'est-à-dire que la validation a été réussie).

L'exception **ActivityInProgress** est déclenchée quand l'activité logicielle a été lancée et ne peut pas être annulée.

L'exception **CommFailure** est déclenchée en cas de panne du RCD entre le système de gestion par fournisseur et la terminaison OLT ou de panne de communication entre terminaison OLT et terminaison ONT de départ.

L'exception **DeniedAccess** est déclenchée si l'accès à l'élément de réseau est refusée en conséquence de restrictions relatives au contrôle d'accès.

L'exception **InstallationFailure** est déclenchée si le processus d'installation des logiciels a échoué.

L'exception **InsufficientMemory** est déclenchée s'il n'y a pas assez de mémoire sur l'élément de réseau afin de charger les logiciels.

L'exception **InvalidSoftwareTrackingObject** est déclenchée si l'objet de suivi logiciel cité en référence n'est pas le plus récemment associé à l'installation d'une charge logicielle sur l'élément de réseau.

L'exception **InvalidStartTime** est déclenchée si l'instant de début est antérieur à l'instant actuel du système.

L'exception **SoftwareLoadHWMismatch** est déclenchée si le logiciel désigné ne peut pas être chargé sur le matériel de l'équipement étant donné que la version du matériel ne peut pas accepter cette charge logicielle.

L'exception **SoftwareNotYetInstalled** est déclenchée quand l'activation est requise et que les logiciels n'ont pas encore été installés.

L'exception **SoftwareTrackingObjectInUse** est déclenchée quand il y a en instance des activités logicielles suivies par cet objet et que ces activités ne peuvent pas être supprimées.

L'exception **SourceUnreachable** est déclenchée si le serveur détenant la charge logicielle à télécharger n'a pas pu être atteint par la terminaison OLT.

L'exception **Timeout** est déclenchée si la durée de traitement a atteint une limite de temporisation définie par le système avant que le processus puisse s'achever.

L'exception **UnknownSoftwareLoad** est déclenchée quand l'ensemble logiciel spécifié ne peut pas être trouvé.

L'exception **UnknownSoftwareDownloadTrackingObject** est déclenchée quand le processus de téléchargement du logiciel est inconnu du système de gestion par fournisseur.

L'exception **UnrecognizedTarget** est déclenchée quand le logiciel désigné dans le serveur de fichiers sécurisés est inconnu du système de gestion par fournisseur.

9.5.2 Interface VersionRepository

9.5.2.1 retrieveVersions

Cette opération extrait toutes les informations de version (aussi bien logicielle que matérielle) pour une ressource réseau.

La signature de l'opération **retrieveVersions** est indiquée ci-dessous:

```
VersionsSeqType retrieveVersions (
                              in ManagedEntityIdType containingManagedEntityId)
                              raises (CommFailure, UnknownManagedEntity,
                              AccessDenied);
```

Le paramètre d'entrée **containingManagedEntityId** désigne la ressource réseau.

La valeur de retour est de type **VersionsSeqType** et indique une énumération des versions matérielles et logicielles associées à cette ressource réseau.

9.5.2.2 validateNEVersion

Cette opération demande au système de gestion par fournisseur de valider si le logiciel proposé est compatible avec l'élément de réseau.

La signature de l'opération **validateNEVersion** est indiquée ci-dessous:

```
boolean validateNEVersion (
            in ManagedEntityIdType managedEntityId,
            in VersionType proposedSoftware)
            raises (UnknownNE, AccessDenied);
```

Le paramètre d'entrée **managedEntityId** désigne l'élément de réseau.

Le paramètre d'entrée **proposedSoftware** désigne le logiciel dont la validation est proposée.

La valeur de retour est de type **booléen**.

9.5.2.3 validatePlugInVersion

Cette opération demande au système de gestion par fournisseur de valider si le logiciel proposé est compatible avec le module d'extension.

La signature de l'opération **validatePlugInVersion** est indiquée ci-dessous:

```
boolean validatePlugInVersion (
                  in ManagedEntityIdType plugInUnitId,
                  in VersionType proposedSoftware)
                  raises (UnknownManagedEntity, AccessDenied);
```

Le paramètre d'entrée **plugInUnitId** désigne le module d'extension.

Le paramètre d'entrée **proposedSoftware** désigne le logiciel dont la validation est proposée.

La valeur de retour est de type **booléen**.

9.5.2.4 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **CommFailure** est déclenchée en cas de panne du RCD entre système de gestion par fournisseur et terminaison OLT ou de panne de communication entre terminaison OLT et terminaison ONT de départ.

L'exception **UnknownManagedEntity** est déclenchée si l'identificateur du module d'extension ou de l'accès est inconnu du système de gestion par fournisseur.

9.6 Module EventPublisher

Dès réception d'informations traitées concernant la configuration, la qualité de fonctionnement ou un événement de dérangement, fournies par d'autres scénarios dans le système de gestion par fournisseur et fondées sur des règles concernant la publication, le système de gestion par fournisseur met en file d'attente et canalise les informations relatives aux événements vers tous les consommateurs intéressés, y compris les opérateurs et le ou les systèmes OMS.

9.6.1 Interface AlarmEventSupplier

Le rôle de cette interface est d'annoncer des événements d'alarme au système de gestion par opérateur au moyen du service de notification du groupe OMG. Cette interface ne comporte pas d'opérations. Cependant, elle fournit bien le mappage d'en-tête fixe ainsi que les mappages de données filtrables pour l'objet d'événement structuré qui sert à communiquer des informations relatives aux événements au moyen du canal d'événements du service de notification du groupe OMG. Ces deux ensembles de mappages doivent suivre les lignes directrices spécifiées dans la Rec. UIT-T X.780 pour l'interface de notifications.

Dans l'en-tête fixe, le type **domain_type** est mis à la valeur "télécommunications", le nom **type_name** est mis à la valeur "Alarme", et le nom **event_name** est mis à une des valeurs suivantes d'une chaîne constante: "Communications Alarm", "Environmental Alarm", "Equipment Alarm", "Processing Error Alarm", ou "Quality of Service Alarm".

Le mappage dans les données filtrables se compose de paires d'éléments. Le premier composant contenu dans la paire est un identificateur de chaîne pour un nom de données et le second est la valeur de cet élément de données. Les identificateurs de chaîne sont des constantes qui sont définies dans cette interface. Par ailleurs, les paires de données filtrables doivent toujours apparaître dans un ordre spécifique.

L'ordre des éléments filtrables est le suivant:

- AlarmEmittingMEId
- EventTime
- ProbableCause
- SpecificProblems
- PerceivedSeverity
- ServiceAffectingInd
- BackUpStatus
- BackUpManagedEntityId
- ThresholdInfo
- NotificationIdentifier
- CorrelatedNotifications
- StateChangeDefinition

- AdditionalText

La valeur de l'élément ProbableCause a la syntaxe du type ProbableCauseType et adopte une des valeurs spécifiques définies dans la Rec. UIT-T X.780 ou q834_4::Q834Common::ProbableCauseConst.

La valeur de l'élément ThresholdInfo a la syntaxe du type MonitoredParameterType et adopte une des valeurs spécifiques définies dans le paramètre q834_4::Q834Common::MonitoredParameter. La chaîne vide est fournie dans cette valeur pour tous les noms d'événement sauf "QualityOfServiceAlarm".

La syntaxe et l'interprétation de tous les autres types de données et l'interprétation de leur utilisation sont indiquées dans la Rec. UIT-T X.780.

9.6.2 SecurityEventSupplier

Le rôle de cette interface est d'annoncer des événements de sécurité au système de gestion par opérateur au moyen du service de notification du groupe OMG. Cette interface ne comporte pas d'opérations. Cependant, elle fournit bien le mappage d'en-tête fixe ainsi que les mappages de données filtrables pour l'objet d'événement structuré servant à communiquer des informations relatives aux événements au moyen du canal d'événements du service de notification du groupe OMG. Ces deux ensembles de mappages doivent suivre les lignes directrices spécifiées dans la Rec. UIT-T X.780 pour l'interface de notifications.

Dans l'en-tête fixe, le type **domain_type** est mis à la valeur "télécommunications", le nom **type_name** est mis à la valeur "SecurityEvent", et le nom **event_name** est mis à la valeur d'une chaîne constante qui a une des valeurs suivantes: "IntegrityViolation", "OperationalViolation", "PhysicalViolation", "SecurityEventViolation" ou "TimeDomainViolation".

Le mappage dans les données filtrables se compose de paires d'éléments. Le premier composant contenu dans la paire est un identificateur de chaîne pour un nom de données et le second est la valeur de cet élément de données. Les identificateurs de chaîne sont des constantes qui sont définies dans cette interface. Par ailleurs, les paires de données filtrables doivent toujours apparaître dans un ordre spécifique.

L'ordre des éléments filtrables est le suivant:

- EventEmittingMEId
- EventTime
- SecurityAlarmCause
- SecurityAlarmDetector
- ServiceUser
- ServiceProvider
- NotificationIdentifier
- CorrelatedNotifications
- AdditionalText

La syntaxe des types de données et l'interprétation de leur utilisation se trouvent dans la Rec. UIT-T X.780.

9.6.3 DiscoveryEventSupplier

Le rôle de cette interface est d'annoncer des modifications des équipements installés au système de gestion par opérateur au moyen du service de notification du groupe OMG. Cette interface ne comporte pas d'opérations. Cependant, elle fournit bien le mappage d'en-tête fixe ainsi que les mappages de données filtrables pour l'objet d'événement structuré servant à communiquer des

informations relatives aux événements au moyen du canal d'événements du service de notification du groupe OMG.

Dans l'en-tête fixe, le type **domain_type** est mis à la valeur "télécommunications", le nom **type_name** est mis à la valeur "DiscoveryEvent", et le nom **event_name** est mis à la valeur d'une chaîne constante qui a une des valeurs suivantes: "ManagedEntityCreation" ou "ManagedEntityDeletion".

Le mappage dans les données filtrables se compose de paires d'éléments. Le premier composant contenu dans la paire est un identificateur de chaîne pour un nom de données et le second est la valeur de cet élément de données. Les identificateurs de chaîne sont des constantes qui sont définies dans cette interface. Par ailleurs, les paires de données filtrables doivent toujours apparaître dans un ordre spécifique.

L'ordre des éléments filtrables pour un nom event_name de valeur "ManagedEntityCreation" est le suivant:

- ManagedEntityType
- EventTime
- ManagedEntityAttributeValues
- NotificationIdentifier
- CorrelatedNotifications
- AdditionalText

La valeur de l'élément ManagedEntityType a la syntaxe du type EquipmentType qui indique le type de données d'inventaire découvertes. L'interface définit des constantes pour divers types d'élément de réseau ainsi que pour divers modules d'extension, logements d'équipement et logiciels.

La valeur de l'élément EventTime a la syntaxe du type GeneralizedTimeType et se rapporte au moment où l'état de découverte a été détecté par le réseau.

La valeur de l'élément ManagedEntityAttributeValues a la syntaxe MEstruct. Cependant, le type de données fourni pour cette valeur fait partie d'un ensemble de structures de données défini dans le module. L'élément de données ManagedEntityType désigne le type de structure qui est transmis dans cette valeur par l'objet d'événement structuré.

La valeur de l'élément NotificationIdentifier a la syntaxe du type NotificationIdentifierType. Elle fournit un numéro séquentiel de référence pour l'événement. La valeur de l'élément CorrelatedNotifications a la syntaxe du type CorrelatedNotificationType et fournit une liste de numéros de référence pour d'autres notifications d'événement fournies par le système de gestion du fournisseur pour des conditions d'inventaire associées. S'il n'y a aucune notification associée, la valeur de l'ensemble vide est fournie.

Finalement, la valeur de l'élément AdditionalText a la syntaxe d'une chaîne. Cet élément de données indique un emplacement afin de transmettre d'éventuelles informations diverses en mode texte issues du système de gestion par fournisseur concernant l'état de changement d'inventaire. S'il n'y a pas d'informations additionnelles, la chaîne vide sera transmise.

L'ordre des éléments filtrables pour un nom d'événement de valeur "ManagedEntityDeletion" est le suivant:

- ManagedEntityType
- EventTime
- ManagedEntityAttributeValues
- NotificationIdentifier
- CorrelatedNotifications

- AdditionalText

La valeur de l'élément ManagedEntityType a la syntaxe du type EquipmentType qui indique le type de données d'inventaire découvertes. L'interface définit des constantes pour divers types d'élément de réseau ainsi que pour divers modules d'extension et logements d'équipement.

La valeur de l'élément EventTime a la syntaxe du type GeneralizedTimeType et se rapporte au moment où l'état de suppression a été détecté par le réseau.

La valeur de l'élément ManagedEntityAttributeValues a la syntaxe MEstruct. Cependant, le type de données fourni pour cette valeur fait partie d'un ensemble de structures de données défini dans le module. L'élément de données ManagedEntityType désigne le type de structure qui est transmis dans cette valeur par l'objet d'événement structuré.

La valeur de l'élément NotificationIdentifier a la syntaxe NotificationIdentifierType. Elle fournit un numéro séquentiel de référence pour l'événement. La valeur de l'élément CorrelatedNotifications a la syntaxe du type CorrelatedNotificationType et fournit une liste de numéros de référence pour d'autres notifications d'événement fournies par le système de gestion du fournisseur pour des conditions de changement d'inventaire associées. S'il n'y a aucune notification associée, la valeur de l'ensemble vide est fournie.

Finalement, la valeur de l'élément AdditionalText a la syntaxe de chaîne. Cet élément de données indique un emplacement afin de transmettre d'éventuelles informations diverses en mode texte issues du système de gestion par fournisseur concernant l'état de changement d'inventaire. S'il n'y a pas d'informations additionnelles, la chaîne vide sera transmise.

L'insertion ou la suppression d'un module d'extension crée toujours une notification, quel que soit l'état de préfourniture qui entoure ce module d'extension.

9.7 Module MIBTransfer

Ce module gère le processus d'acquisition de données de configuration du système à utiliser afin de restaurer un système au cas où un état de défaillance cataleptique se serait produit. L'acquisition des données de configuration peut être effectuée soit en temps réel ou en temps planifié. Ce module indique également des informations d'état relatives au processus de sauvegarde et de restauration en cours. Les résultats des processus de sauvegarde et de restauration sont journalisés par le système de gestion du fournisseur. Toute demande de sauvegarde ou de restauration des données de configuration est accompagnée de justificatifs de sûreté par lesquels le système de gestion par fournisseur ou la terminaison OLT est autorisé à communiquer avec le serveur externe. Pendant que la sauvegarde est en cours d'avancement, toutes les demandes de fourniture mettant en jeu le système de terminaison OLT désigné sont rejetées/bloquées. Les objets de suivi de transfert sont automatiquement supprimés par le système de gestion du fournisseur une fois que le ou les transferts associés de fichiers sont terminés et les résultats (échec ou succès) sont enregistrés dans le journal d'exécution.

9.7.1 Interface MIBMover

9.7.1.1 startBackup

Cette opération lance une sauvegarde immédiate des données de configuration de système à partir d'un système et/ou du système de gestion par fournisseur vers une destination de serveur de sauvegarde.

La signature de l'opération **startBackup** est indiquée ci-dessous:

```
TransferTrackingObjectIdType startBackup (
                              in managedEntityIdType nEManagedEntityId,
                              in DCNAddressType destinationServerAddr,
                              in UserIdType userId,
```

```
                                in PasswordType password,
                                in FilenameType destinationFile,
                                in boolean overwriteExistingFile)
                                raises (UnknownNE, UnknownDestinationServer,
                                CommFailure, EquipmentFailure, DeniedAccess,
                                AccessDenied);
```

Le paramètre d'entrée **nEManagedEntityId** désigne le système à sauvegarder. Le paramètre d'entrée **destinationServerAddr** désigne l'adresse de mise en réseau de communication de données (RCD) pour le serveur qui est la destination de la sauvegarde. Le paramètre d'entrée **userId** désigne l'identifiant d'utilisateur auprès du serveur de destination. Le paramètre d'entrée **password** est le mot de passe permettant d'accéder au serveur de destination. Le paramètre d'entrée **destinationFile** indique un répertoire d'accueil entier pour le fichier de sauvegarde. Finalement, le paramètre **overwriteExistingFile** indique si la sauvegarde devrait permettre l'écrasement d'un fichier préexistant ayant le même emplacement de répertoire d'accueil.

La valeur de retour est de type **TransferTrackingObjectIdType** et indique une clé de corrélation permettant de suivre l'état d'avancement du processus de sauvegarde.

9.7.1.2 getBackupStatus

Cette opération offre la capacité d'extraire l'état d'un processus de sauvegarde.

La signature de l'opération **getBackupStatus** est indiquée ci-dessous:

```
StatusAttributeSeqType getBackupStatus (
                        in TransferTrackingObjectIdType id)
                        raises (UnknownBackupProcess,AccessDenied);
```

Le paramètre d'entrée **id** désigne le processus de sauvegarde.

La valeur de retour est de type **StatusAttributeSeqType** et indique l'état du processus de sauvegarde déjà requis.

9.7.1.3 scheduleBackup

Cette opération planifie les processus de sauvegarde.

La signature de l'opération **scheduleBackup** est indiquée ci-dessous:

```
TransferTrackingObjectIdType scheduleBackup(
                             in ManagedEntityIdType nEManagedEntityId,
                             in UserIdType userId,
                             in PasswordType   password,
                             in UserLabelType schedulerName,
                             in DCNAddressType destinationServerAddr,
                             in FilenameType destinationFile,
                             in boolean overwriteExistingFile)
                             raises (UnknownNE, UnknownScheduler,
                             UnknownDestinationServer, InvalidScheduler,
                             AccessDenied);
```

Le paramètre d'entrée **nEManagedEntityId** désigne la terminaison OLT à sauvegarder. Le paramètre d'entrée **schedulerName** est le nom du programmateur à utiliser pour la sauvegarde. Le paramètre d'entrée **userId** désigne l'identifiant d'utilisateur auprès du serveur de destination. Le paramètre d'entrée **password** est le mot de passe permettant d'accéder au serveur de destination. Le paramètre **destinationServerAddr** indique l'adresse du réseau de communication de données (RCD) pour le serveur de destination où les données sont à sauvegarder. Le paramètre d'entrée **destinationFile** indique un répertoire d'accueil entier pour le fichier de sauvegarde. Finalement, le paramètre **overwriteExistingFile** indique si la sauvegarde devrait permettre l'écrasement d'un fichier préexistant ayant le même emplacement de répertoire d'accueil.

La valeur de retour est de type **TransferTrackingObjectIdType** et indique une clé de corrélation à utiliser lors d'une tentative ultérieure de suivi de l'état du temps planifié.

9.7.1.4 modifyBackupSchedule

Cette opération annule tous les processus de sauvegarde ultérieurs pour un système fondé sur un programmateur. Cette opération n'interrompra pas un processus de sauvegarde en cours.

La signature de l'opération **modifyBackupSchedule** est indiquée ci-dessous:

```
void modifyBackupSchedule (
              in TransferTrackingObjectIdType id,
              in UserLabelType newSchedulerName)
              raises (UnknownBackupProcess, AccessDenied,
              UnknownScheduler, InvalidScheduler);
```

Le paramètre d'entrée **id** désigne le processus planifié à modifier. Le paramètre d'entrée **newSchedulerName** désigne les nouveaux déclencheurs temporels.

La valeur de retour est de type **void**.

9.7.1.5 cancelScheduledBackup

Cette opération annule tous les processus de sauvegarde ultérieurs pour un système fondé sur un programmateur. Cette opération n'interrompra pas un processus de sauvegarde en cours.

La signature de l'opération **cancelScheduledBackup** est indiquée ci-dessous:

```
void cancelScheduledBackup (
            in TransferTrackingObjectIdType id)
            raises (UnknownBackupProcess, AccessDenied);
```

Le paramètre d'entrée **id** désigne le processus planifié à annuler.

La valeur de retour est de type **void**.

9.7.1.6 abortBackup

Cette opération abandonne un processus de sauvegarde en cours, planifié ou non. Les processus de sauvegarde planifiés ultérieurement ne sont pas affectés par cette opération.

La signature de l'opération **abortBackup** est indiquée ci-dessous:

```
void abortBackup (
            in TransferTrackingObjectIdType id)
            raises (UnknownBackupProcess, CommFailure, EquipmentFailure,
            AccessDenied);
```

Le paramètre d'entrée **id** désigne le processus de sauvegarde à abandonner.

La valeur de retour est de type **void**.

9.7.1.7 startRestore

L'opération **startRestore** offre la capacité de rétablir un système sur la base d'une copie sauvegardée des données de configuration.

La signature de l'opération **startRestore** est indiquée ci-dessous. Les données de configuration résident dans un serveur externe sécurisé.

```
TransferTrackingObjectIdType startRestore (
                              in ManagedEntityIdType nEManagedEntityId,
                              in DCNAddressType sourceServerAddr,
                              in UserIdType userId,
                              in PasswordType password,
                              in FilenameType sourceFile)
```

```
                                raises (UnknownNE, CommFailure,
                                EquipmentFailure, UnknownSourceServer,
                                DeniedAccess, SoftwareLoadHardwareMismatch,
                                AccessDenied );
```

Le paramètre d'entrée **nEManagedEntityId** désigne la terminaison OLT à restaurer. Le paramètre d'entrée **sourceServerAddr** désigne l'adresse de mise en réseau de communication de données (RCD) pour le serveur qui est la destination de la sauvegarde. Le paramètre d'entrée **userId** désigne l'identifiant d'utilisateur auprès du serveur de destination. Le paramètre d'entrée **password** est le mot de passe permettant d'accéder au serveur de destination. Le paramètre **sourceFile** indique le fichier où ont été sauvegardées les données dont la restauration sera effectuée.

La valeur de retour est de type **TransferTrackingObjectIdType** et indique une clé de corrélation à utiliser lors d'une tentative de suivi de l'état de la restauration requise.

9.7.1.8 getRestoreStatus

Cette opération indique l'état d'un processus de restauration.

La signature de l'opération **getRestoreStatus** est indiquée ci-dessous:

```
StatusAttributeSeqType getRestoreStatus (
                        in TransferTrackingObjectIdType id)
                        raises (UnknownRestoreProcess, AccessDenied);
```

Le paramètre d'entrée **id** désigne le processus de restauration.

La valeur de retour est de type **StatusAttributeSeqType** et indique l'état d'avancement du processus de restauration.

9.7.1.9 transferTrackingObjectIdListGet

Cette opération extrait la liste des transferts en instance dans la base MIB du système, à la fois de sauvegarde et de restauration.

La signature de l'opération **transferTrackingObjectIdListGet** est indiquée ci-dessous:

```
TransferTrackingObjectIdSeqType transferTrackingObjectIdListGet ()
                    raises (AccessDenied);
```

Il n'y a pas de paramètre d'entrée.

La valeur de retour est de type **TransferTrackingObjectIdSeqType** et indique la liste de transferts en instance dans la base MIB du système, à la fois de sauvegarde et de restauration.

9.7.1.10 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **CommFailure** est déclenchée en cas de panne du RCD entre système de gestion par fournisseur et terminaison OLT ou de panne de communication entre terminaison OLT et terminaison ONT de départ.

L'exception **DeniedAccess** est déclenchée si l'accès à l'élément de réseau est refusé.

L'exception **EquipmentFailure** est déclenchée quand l'équipement duquel les données sont sauvegardées se trouve en état de panne.

L'exception **InvalidScheduler** est déclenchée quand le programmateur indiqué ne convient pas pour utilisation dans cette opération ou est périmé.

L'exception **SoftwareLoadHardwareMismatch** est déclenchée quand le logiciel de départ ne concorde pas avec le matériel de destination.

L'exception **UnknownBackupProcess** est déclenchée quand le processus de sauvegarde associé à l'identificateur TransferTrackingObjectId indiqué n'est pas trouvé.

L'exception **UnknownDestinationServer** est déclenchée quand l'adresse IP du serveur de destination n'est pas trouvée.

L'exception **UnknownNE** est déclenchée quand la terminaison OLT est inconnue du système de gestion par fournisseur.

L'exception **UnknownRestoreProcess** est déclenchée quand le processus de restauration associé à l'identificateur TransferTrackingObjectId est inconnu.

L'exception **UnknownScheduler** est déclenchée quand le nom indiqué du programmateur n'est pas trouvé.

L'exception **UnknownSourceServer** est déclenchée quand le serveur de départ est inconnu du système de gestion par fournisseur.

9.8 Module PerformanceManager

Le système de gestion par fournisseur doit autoriser l'activation et la désactivation de la signalisation de comptes rendus des données de qualité de fonctionnement ou des mesures de trafic relevées sur des terminaisons individuelles contenues dans les éléments de réseau comme requis par l'opérateur ou par le système OMS pour les terminaisons OLT qui ont été installées. Les exigences applicables au système de gestion par fournisseur, figurant dans la présente Recommandation, comprennent également une disposition relative au réglage des valeurs de seuil. Elles décrivent également la signalisation de comptes rendus automatique de mesures de qualité de fonctionnement au système de gestion par fournisseur quand des seuils ont été franchis. En l'occurrence d'un ensemble d'alertes de débordement de seuil associé à une unique situation de dégradation de qualité, le système de gestion par fournisseur doit analyser et corréler les événements d'alerte dans son domaine au mieux de sa capacité, déterminer la cause fondamentale sous-jacente du problème, et mémoriser ces informations dans un journal. Si plusieurs occurrences de la même dégradation à cause fondamentale sont détectées dans un intervalle de temps, le système de gestion par fournisseur doit préparer un enregistrement d'alarme de QS pour publication auprès de tout consommateur intéressé (l'opérateur ou le système OMS). Si le système de gestion par fournisseur peut recueillir tous les enregistrements de données chronologiques pour tous les points de surveillance[5] de tous les éléments de réseau dans son domaine de gestion, il n'est pas nécessaire de contrôler la signalisation. Dans ce cas, l'on part du principe que la signalisation de comptes rendus est constamment disponible.

9.8.1 Interface ImpairmentPersistence

Dans l'ensemble de la présente Recommandation, il est supposé et requis que les objets de profil qui sont des données de type seuil[6] soient des groupements de valeurs de seuil pouvant être associées à un et un seul type de point de surveillance.

9.8.1.1 setSlidingWindowParameters

Cette opération règle les paramètres de fenêtre mobile pour un, quelques-uns ou la totalité des paramètres surveillés dans un élément de réseau. Ce réglage est appliqué à tous les points de surveillance des paramètres. Ultérieurement, le système de gestion par fournisseur produira une alarme de QS si un point de surveillance détecte une alarme TCA **persistenceMinimum** fois dans

---

[5] Les enregistrements de données chronologiques et les points de surveillance sont définis dans la Rec. UIT-T Q.834.1.

[6] De type 21 comme spécifié dans le module *ProfileManager*.

totConsecutiveIntvls intervalles de surveillance consécutifs. Cette opération est effectuée au mieux.

L'exception CommFailure est déclenchée si le système de gestion par fournisseur n'est pas capable de communiquer avec l'élément de réseau désigné dans l'opération et si les valeurs paramétriques de fenêtre mobile sont réglées directement en fonction des ressources de réseau.

Cette opération sert à établir des réglages par défaut à l'échelle du système si l'indicateur sysScopeInd a la valeur TRUE et si l'identificateur nEManagedEntityId désigne une terminaison OLT. En d'autres termes, les valeurs paramétriques de fenêtre mobile sont à appliquer à tous les points de surveillance de tous les composants d'équipement appartenant actuellement ou pouvant appartenir au système avec la tête de réseau nommée de la terminaison OLT. Si la terminaison OLT a perdu ses communications avec une ressource réseau sous-jacente avant ou pendant l'invocation de cette opération, c'est à l'implémentation du fournisseur qu'il appartient de garantir que les réglages sont appliqués une fois que la communication est établie. Si l'indicateur sysScopeInd a la valeur TRUE et que l'identificateur neManagedEntityId désigne une terminaison ONU, alors les réglages sont considérés comme s'appliquant à la terminaison ONU et à toutes les terminaisons de réseau qui la sous-tendent. Lorsque des composants d'équipement sont ajoutés au système avec la tête de réseau de la terminaison OLT ou ONU, le système de gestion par fournisseur règle automatiquement ces valeurs paramétriques.

Si l'indicateur sysScopeInd a la valeur FALSE, alors les réglages paramétriques de fenêtre mobile sont à appliquer à la seule ressource indiquée. Ces réglages restent appliqués aux composants d'équipement du nœud qui existent ou peuvent exister.

La signature de l'opération **setSlidingWindowParameters** est indiquée ci-dessous:

```
void setSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in MonitoredParameterSeqType monitoredParameterList,
            in short totConsecutiveIntvls,
            in short persistenceMinimum,
            in boolean sysScopeInd)
            raises (UnknownNE, UnknownParameters, IntervalCountTooLarge,
            AccessDenied, CommFailure);
```

Le paramètre d'entrée **nEManagedEntityId** désigne l'élément de réseau. Le paramètre d'entrée **monitoredParameterList** désigne les paramètres à surveiller. Les valeurs des paramètres sont indiquées dans q834_4::Q834Common::MonitoredParameter. Le paramètre d'entrée **totConsecutiveIntvls** désigne la valeur du nombre total d'intervalles surveillés consécutifs. Le paramètre d'entrée **persistenceMinimum** désigne le nombre minimal d'occurrences de l'alarme TCA pour le paramètre surveillé. L'entrée **sysScopeInd** indique si les réglages de fenêtre mobile devraient être appliqués à tous les éléments de réseau sous-jacents à partir de l'élément initial qui est désigné dans l'opération.

La valeur de retour est de type **void**.

9.8.1.2 setSpecificSlidingWindowParameters

Cette opération est semblable à l'opération setSlidingWindowParameters sauf que le domaine d'attribution des réglages est limité à un point de surveillance spécifiquement désigné dans l'opération.

La signature de l'opération **setSpecificSlidingWindowParameters** est indiquée ci-dessous:

```
void setSpecificSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in MonitoredParameterSeqType monitoredParameterList,
            in short totConsecutiveIntvls,
```

```
            in short persistenceMinimum,
            in boolean allowGlobalOverwrite)
            raises (UnknownNE, UnknownParameters, IntervalCountTooLarge,
            AccessDenied, UnknownManagedEntity, CommFailure, EquipmentFailure);
```

Le paramètre d'entrée **nEManagedEntityId** désigne l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **monitoringPoint** désigne le point spécifique de surveillance afin de régler les paramètres de fenêtre mobile pour un paramètre surveillé. Le paramètre d'entrée **monitoredParameterList** désigne les paramètres à surveiller, dont les valeurs sont définies dans Q834Common::MonitoredParameter. Le paramètre d'entrée **totConsecutiveIntvls** désigne la valeur totale d'intervalles surveillés consécutifs. Le paramètre d'entrée **persistenceMinimum** désigne le nombre minimal d'occurrences de l'alarme TCA pour le paramètre surveillé. Le paramètre **allowGlobalOverwrite** indique si les paramètres spécifiques de fenêtre mobile peuvent être écrasés lors d'une invocation ultérieure de l'opération **setSlidingWindowParameters**.

La valeur de retour est de type **void**.

9.8.1.3 getSpecificSlidingWindowParameters

Cette opération énumère des points de surveillance et de leurs réglages de fenêtre mobile pour un paramètre spécifié.

La signature de l'opération **getSpecificSlidingWindowParameters** est indiquée ci-dessous:

```
SWPValueSeqType getSpecificSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in MonitoredParameterType monitoredParameter)
            raises (UnknownNE, UnknownParameters, CommFailure);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **monitoredParameter** désigne le paramètre à surveiller et est défini dans q834_4::Q834Common::MonitoredParameter.

La valeur de retour est de type **SWPValueSeqType** et indique les attributions de fenêtre mobile pour un point de surveillance spécifique.

9.8.1.4 setThreshold

Cette opération règle la valeur de seuil désignée par un profil à un point de surveillance dans un élément de réseau.

La signature de l'opération **setThreshold** est indiquée ci-dessous:

```
void setThreshold (
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in NameType thresholdDataProfileName )
            raises (UnknownNE, AccessDenied,UnknownManagedEntity,
            UnknownProfiles, InvalidAssociation, CommFailure, ProfileSuspended);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **monitoringPoint** désigne le point spécifique de surveillance afin de régler les paramètres de fenêtre mobile pour un paramètre surveillé. Le paramètre d'entrée **thresholdDataProfileName** énumère des noms de profil pour des valeurs de seuil.

La valeur de retour est de type **void**.

9.8.1.5 setThresholds

Cette opération permet le réglage d'un ensemble de paires de valeurs de seuil et de type de point de surveillance sur un élément de réseau particulier. L'exception CommFailure est déclenchée si le

système de gestion par fournisseur n'est pas capable de communiquer avec l'élément de réseau désigné dans l'opération et si les valeurs de seuil sont réglées directement sur les ressources de réseau.

Cette opération sert à établir des réglages par défaut à l'échelle du système si l'indicateur sysScopeInd a la valeur TRUE et si nEManagedEntityId est l'identificateur d'une terminaison OLT. En d'autres termes, les valeurs de seuil sont à appliquer à tous les points de surveillance de tous les composants d'équipement appartenant actuellement ou pouvant appartenir au système avec la tête de réseau nommée de la terminaison OLT. Si la terminaison OLT a perdu ses communications avec une quelconque ressource réseau sous-jacente avant ou pendant l'invocation de cette opération, c'est à l'implémentation du fournisseur qu'il appartient de garantir que les réglages sont appliqués une fois que communication est établie. Si l'indicateur sysScopeInd a la valeur TRUE et que neManagedEntityId soit l'identificateur d'une terminaison ONU, alors les réglages sont considérés comme s'appliquant à la terminaison ONU et à toutes les terminaisons de réseau qui la sous-tendent. Lorsque des composants d'équipement sont ajoutés au système avec tête de réseau de la terminaison OLT ou ONU, le système de gestion par fournisseur règle automatiquement ces valeurs de seuil.

Si l'indicateur sysScopeInd a la valeur FALSE, alors les réglages de seuil sont à appliquer seulement à la ressource indiquée. Les réglages sont encore appliqués aux composants d'équipement du nœud, qu'il soient existants ou potentiels.

La signature de l'opération **setThresholds** est indiquée ci-dessous:

```
void setThresholds (
                        in ManagedEntityIdType nEManagedEntityId,
                        in boolean sysScopeInd,
                        in ThresholdsSeqType thresholdsList)
                        raises (UnknownNE, UnknownProfiles,
                        AccessDenied,UnknownMonitoringPointTypes,
                        InvalidAssociation, CommFailure, ProfileSuspended );
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **thresholdList** désigne une liste de paires se composant d'un nom de données de seuil et d'un type de point de surveillance. L'entrée **sysScopeInd** indique si les réglages de fenêtre mobile devraient être appliqués à tous les éléments de réseau sous-jacents de l'élément initial qui est désigné dans l'opération.

La valeur de retour est de type **void**.

9.8.1.6 getThresholdValues

Cette opération énumère les points de surveillance et leurs noms de réglage des données de seuil pour un type spécifié de point de surveillance.

La signature de l'opération **getThresholdValues** est indiquée ci-dessous:

```
MonitoringPointThresholdsSeqType getThresholdValues (
               in ManagedEntityIdType nEManagedEntityId,
               in MonitoringPointKindType monitoringPointType)
               raises (UnknownNE, UnknownMonitoringPointTypes, CommFailure);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **monitoringPointType** désigne le point de surveillance sur lequel l'extraction est fondée.

La valeur de retour est de type **MonitoringPointThresholdsSeqType** et indique les valeurs de réglage pour les franchissements de seuil.

9.8.1.7 getSystemThresholdsSetting

Cette opération extrait les valeurs par défaut du système pour les données de seuil.

La signature de l'opération **getSystemThresholdsSetting** est indiquée ci-dessous:

```
ThresholdsSeqType getSystemThresholdsSetting (
                  in ManagedEntityIdType nEManagedEntityId)
                  raises (AccessDenied, UnknownManagedEntity);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique le système pour lequel des réglages par défaut sont requis.

La valeur de retour est de type **ThresholdsSeqType** et indique la liste des valeurs par défaut du système.

9.8.1.8 getSystemSWSettings

Cette opération extrait les réglages par défaut du système concernant la fenêtre mobile.

La signature de l'opération **getSystemSWSettings** est indiquée ci-dessous:

```
ParameterSettingSeqType getSystemSWSettings (
                  in ManagedEntityIdType nEManagedEntityId)
                  raises (AccessDenied, UnknownManagedEntity);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique le système pour lequel des réglages par défaut de fenêtre mobile sont requis.

La valeur de retour est de type **ParameterSettingSeqType** et indique la liste des valeurs par défaut du système.

9.8.1.9 Exceptions

L'exception **AccessDenied** est déclenchée quand le système de gestion par opérateur n'est pas autorisé à accéder à cet objet d'interface.

L'exception **CommFailure** est déclenchée quand il y a eu une panne de liaison RCD entre les éléments de réseau et que le système de gestion par fournisseur a une défaillance.

L'exception **EquipmentFailure** est déclenchée quand l'équipement est en état de panne et que les réglages ne peuvent pas être appliqués.

L'exception **InvalidAssociation** est déclenchée quand le profil indiqué ne peut pas être appliqué à un point de surveillance.

L'exception **IntervalCountTooLarge** est déclenchée quand les intervalles requis dépassent la durée maximale prise en charge par le système de gestion du fournisseur. Cette exception indique le nombre maximal autorisé d'intervalles de surveillance pris en charge par le système de gestion du fournisseur.

L'exception **ProfileSuspended** est déclenchée quand le ou les profils nommés dans l'invocation ont été suspendus pour utilisation dans le système de gestion fournisseur par le système OMS ou par l'opérateur.

L'exception **UnknownNE** est déclenchée quand l'élément de réseau mentionné dans la demande est inconnu du système de gestion par fournisseur.

L'exception **UnknownManagedEntity** est déclenchée quand le point de surveillance est inconnu du système de gestion par fournisseur.

L'exception **UnknownMonitoringPointTypes** est déclenchée quand le point de surveillance est inconnu du système de gestion par fournisseur.

L'exception **UnknownParameters** est déclenchée quand le paramètre surveillé qui a été indiqué est inconnu dans le système de gestion par fournisseur.

L'exception **UnknownProfiles** est déclenchée si le nom de profil fourni est inconnu du système de gestion par fournisseur et ne peut pas être extrait du répertoire d'objets de profil.

9.8.2 ReportController Interface

9.8.2.1 addCustomerMonitoringReporting

Cette opération ajoute un type de données chronologiques à surveiller à un point de surveillance spécifique en réponse à une réclamation de client désignée par l'identificateur d'instance de service indiqué ou afin de prendre en charge des services de gestion CNM.

La signature de l'opération **addCustomerMonitoringReporting** est indiquée ci-dessous:

```
void addCustomerMonitoringReporting  (
          in ManagedEntityIdType nEManagedEntityId,
          in ServiceInstanceIdType serviceInstanceId,
          in ManagedEntityIdType monitoringPoint,
          in GeneralizedTimeType stopTime,
          in HistoryDataType historyData,
          in short granularityPeriod)
          raises ( UnknownServiceInstance, AccessDenied, UnknownNE,
          UnknownManagedEntity, CollectionPeriodPast,
          CollectionLimitation, InvalidAssociation, UnknownHistoryDataType,
          CommFailure);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **serviceInstanceId** désigne l'instance de service client associée au point de surveillance. Le paramètre d'entrée **monitoringPoint** désigne le point spécifique de surveillance pour l'acquisition de données chronologiques. Le paramètre d'entrée **stopTime** désigne l'instant auquel l'acquisition de données chronologiques va cesser. Si la signalisation de comptes rendus doit être en temps réel, alors stopTime prend la valeur "0". Par ailleurs, la signalisation de comptes rendus se termine automatiquement quand le service est supprimé. Le paramètre d'entrée **historyData** désigne le type de données chronologiques à acquérir, dont les valeurs sont définies dans q834_4::Q834Common::RecordSetType. Le paramètre d'entrée **granularityPeriod** désigne la durée de l'intervalle d'acquisition en minutes. Quand l'instant d'arrêt **stopTime** s'inscrit dans la période de granularité, les données chronologiques sont recueillies pendant l'ensemble de la période de granularité.

La valeur de retour est de type **void**.

9.8.2.2 removeCustomerMonitoringReporting

Cette opération supprime toute acquisition de type de données chronologiques pour le compte d'une instance de service fournie.

La signature de l'opération **removeCustomerMonitoringReporting** est indiquée ci-dessous:

```
void removeCustomerMonitoringReporting  (
              in ServiceInstanceIdType serviceInstanceId )
              raises ( UnknownServiceInstance, AccessDenied,
              CollectionPeriodPast, CommFailure);
```

Le paramètre d'entrée **serviceInstanceId** désigne l'instance de service client associée au point de surveillance.

La valeur de retour est de type **void**.

9.8.2.3 selectByServiceInstance

Cette opération obtient tous les articles enregistrés disponibles dans le système de gestion par fournisseur associé à l'identificateur d'instance de service indiqué. La liste renvoyée n'a aucun

article redondant. Le système de gestion par fournisseur explore toutes les chaînes d'articles correspondantes.

La signature de l'opération **selectByServiceInstance** est indiquée ci-dessous:

```
RecordsSeqType selectByServiceInstance (
              in ServiceInstanceIdType serviceInstanceId,
              in GeneralizedTimeType intervalStartTime,
              in GeneralizedTimeType intervalEndTime)
              raises ( UnknownServiceInstance, AccessDenied);
```

Le paramètre d'entrée **serviceInstanceId** désigne l'instance de service client. Le paramètre d'entrée **intervalStartTime** élimine par filtrage une sélection d'enregistrements de données chronologiques dont l'instant de fin de période se situe avant cette valeur. Le paramètre d'entrée **intervalEndTime** élimine par filtrage une sélection d'enregistrements de données chronologiques dont l'instant de fin de période se situe après cette valeur.

La valeur de retour est de type **RecordsSeqType** et indique une liste d'enregistrements de données chronologiques associés à la surveillance de la qualité de fonctionnement signalée pour cette instance de service.

9.8.2.4 displayActiveReporting

Cette opération obtient tous les points de surveillance pour lesquels des données chronologiques sont actuellement en cours d'acquisition pour un identificateur d'instance de service donné.

La signature de l'opération **displayActiveReporting** est indiquée ci-dessous:

```
MonitoringPointSeqType displayActiveReporting (
              in   ServiceInstanceIdType   serviceInstanceId)
              raises ( UnknownServiceInstance, AccessDenied);
```

Le paramètre d'entrée **serviceInstanceId** désigne l'instance de service client.

La valeur de retour est de type **MonitoringPointSeqType** et indique l'énumération de tous les points de surveillance effectuant un compte rendu actif.

9.8.2.5 addNewMonitoringReporting

Cette opération ajoute un type de données chronologiques à surveiller à un point de surveillance spécifique.

La signature de l'opération **addNewMonitoringReporting** est indiquée ci-dessous:

```
void addNewMonitoringReporting  (
              in ManagedEntityIdType  nEManagedEntityId,
              in   ManagedEntityIdType     monitoringPoint,
              in   GeneralizedTimeType     stopTime,
              in   HistoryDataType    historyData,
              in   short    granularityPeriod)
              raises ( AccessDenied, UnknownNE,
              UnknownManagedEntity, CollectionPeriodPast,
              CollectionLimitation, InvalidAssociation, UnknownHistoryDataType,
              CommFailure);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **monitoringPoint** désigne le point spécifique de surveillance pour l'acquisition de données chronologiques. Le paramètre d'entrée **stopTime** désigne l'instant auquel l'acquisition de données chronologiques va cesser. Si la signalisation de comptes rendus doit être en temps réel, alors l'élément stopTime prend la valeur "0". Par ailleurs, la signalisation de comptes rendus se termine automatiquement quand l'entité gérée (point de surveillance) est supprimée. Le paramètre d'entrée **historyData** désigne le type de données chronologiques à acquérir, dont les valeurs sont définies dans

q834_4::Q834Common::RecordSetType. Le paramètre d'entrée **granularityPeriod** désigne la durée de l'intervalle d'acquisition en minutes. Quand l'instant d'arrêt **stopTime** s'inscrit dans la période de granularité, les données chronologiques sont recueillies pour l'ensemble de la période de granularité.

La valeur de retour est de type **void**.

9.8.2.6 selectByMonitoringPoint

Cette opération obtient tous les articles enregistrés disponibles dans le système de gestion par fournisseur associé au point de surveillance indiqué. La liste renvoyée n'a aucun article redondant.

La signature de l'opération **selectByMonitoringPoint** est indiquée ci-dessous:

```
RecordsSeqType selectByMonitoringPoint (
               in   ManagedEntityIdType     monitoringPoint,
               in   GeneralizedTimeType     intervalStartTime,
               in   GeneralizedTimeType     intervalEndTime)
               raises ( UnknownManagedEntity, AccessDenied);
```

Le paramètre d'entrée **monitoringPoint** désigne la surveillance spécifique pour laquelle des articles enregistrés devraient être extraits. Le paramètre d'entrée **intervalStartTime** élimine par filtrage une sélection d'enregistrements de données chronologiques dont l'instant de fin de période se situe avant cette valeur. Le paramètre d'entrée **intervalEndTime** élimine par filtrage une sélection d'enregistrements de données chronologiques dont l'instant de fin de période se situe après cette valeur.

La valeur de retour est de type **RecordsSeqType** et donne tous les enregistrements de données chronologiques mémorisés dans le système de gestion par fournisseur signalé par le point de surveillance dans la période spécifiée.

9.8.2.7 createReportingSchedule

Cette opération ajoute un ensemble planifié de données chronologiques à surveiller à un point de surveillance spécifique. Le système de gestion par fournisseur recueillera un enregistrement de données chronologiques pour le point de surveillance sur la base du programme de planification. Celui-ci désigne l'instant ou les instants de fin de période pour les articles de journal à acquérir.

La signature de l'opération **createReportingSchedule** est indiquée ci-dessous:

```
void createReportingSchedule  (
               in ManagedEntityIdType  nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in ServiceInstanceIdType serviceInstance,
               in short granularityPeriod,
               in UserLabelType schedulerName)
               raises (AccessDenied, UnknownNE,
               UnknownManagedEntity, CollectionLimitation, UnknownScheduler,
               InvalidAssociation, UnknownHistoryDataType, InvalidScheduler);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **monitoringPoint** désigne le point spécifique de surveillance pour l'acquisition de données chronologiques. Le paramètre d'entrée **historyData** désigne le type de données chronologiques à acquérir, dont les valeurs sont définies dans q834_4::Q834Common::RecordSetType. Le paramètre d'entrée **serviceInstance** fournit une référence facultative à une instance de service. Une valeur de chaîne vide est utilisée si aucune référence n'est recherchée. Le paramètre d'entrée **granularityPeriod** désigne la durée de l'intervalle d'acquisition en minutes. Le paramètre d'entrée **schedulerName** désigne le nom d'un programme de planification qui a été créé antérieurement.

La valeur de retour est de type **void**.

9.8.2.8 modifyReportingSchedule

Cette opération modifie l'acquisition planifiée de données chronologiques à surveiller à un point de surveillance spécifique. En cas de succès, la modification à l'acquisition planifiée de données chronologiques se produira avec l'itération suivante.

La signature de l'opération **modifyReportingSchedule** est indiquée ci-dessous:

```
void modifyReportingSchedule  (
              in ManagedEntityIdType nEManagedEntityId,
              in ManagedEntityIdType monitoringPoint,
              in HistoryDataType historyData,
              in UserLabelType newSchedulerName)
              raises (AccessDenied, UnknownNE,
              UnknownManagedEntity, CollectionLimitation, UnknownScheduler,
              InvalidAssociation, UnknownHistoryDataType, InvalidScheduler);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **monitoringPoint** désigne le point spécifique de surveillance pour l'acquisition de données chronologiques. Le paramètre d'entrée **historyData** désigne le type de données chronologiques à acquérir, dont les valeurs sont définies dans q834_4::Q834Common::RecordSetType. Le paramètre d'entrée **newSchedulerName** désigne le nom du nouveau programme de planification à appliquer.

La valeur de retour est de type **void**.

9.8.2.9 cancelReportingSchedule

Cette opération annule la signalisation planifiée ultérieurement de comptes rendus de données chronologiques lorsque la signalisation de comptes rendus a été requise par l'opérateur antérieurement. Elle n'interrompt pas la signalisation de comptes rendus de données chronologiques en cours ni la signalisation de comptes rendus qui est prise en charge comme comportement par défaut entre le système de gestion par fournisseur et les ressources de réseau.

La signature de l'opération **cancelReportingSchedule** est indiquée ci-dessous:

```
void cancelReportingSchedule (
              in ManagedEntityIdType nEManagedEntityId,
              in ManagedEntityIdType monitoringPoint,
              in HistoryDataType historyData,
              in UserLabelType schedulerName)
              raises (AccessDenied, UnknownNE,
              UnknownManagedEntity, UnknownScheduler,
              InvalidAssociation, UnknownHistoryDataType);
```

Le paramètre d'entrée **nEManagedEntityId** désigne de façon unique l'élément de réseau géré par le système de gestion du fournisseur. Le paramètre d'entrée **monitoringPoint** désigne le point spécifique de surveillance pour l'acquisition de données chronologiques. Le paramètre d'entrée **historyData** désigne le type de données chronologiques à acquérir, dont les valeurs sont définies dans q834_4::Q834Common::RecordSetType. Le paramètre d'entrée **schedulerName** désigne le programme de planification à annuler.

La valeur de retour est de type **void**.

9.8.2.10 Exceptions

L'exception **AccessDenied** est déclenchée quand le système de gestion par opérateur n'est pas autorisé à accéder à cet objet d'interface.

L'exception **CommFailure** est renvoyée quand il y a eu une panne de liaison RCD entre l'élément de réseau et le système de gestion par fournisseur.

L'exception **CollectionLimitation** est déclenchée quand le système de gestion par fournisseur ne peut pas acquérir de données pendant l'intervalle temporel et la période de granularité impartis en raison de restrictions d'implémentation.

L'exception **CollectionPeriodPast** est déclenchée quand l'instant de fin de période est antérieur ou égal à l'heure actuelle.

L'exception **InvalidAssociation** est déclenchée quand le profil indiqué ne peut pas être appliqué à un point de surveillance.

L'exception **InvalidScheduler** est déclenchée quand le nouveau programme de planification est inapproprié pour cette opération ou qu'il est périmé.

L'exception **UnknownNE** est déclenchée quand l'élément de réseau mentionné dans la demande est inconnu du système de gestion par fournisseur.

L'exception **UnknownHistoryDataType** est déclenchée quand le type de données chronologiques est inconnu dans le système de gestion par fournisseur.

L'exception **UnknownManagedEntity** est déclenchée quand le point de surveillance est inconnu du système de gestion par fournisseur.

L'exception **UnknownScheduler** est déclenchée quand le programmateur nommé est inconnu du système de gestion par fournisseur.

L'exception **UnknownServiceInstance** est déclenchée quand l'instance de service est inconnue du système de gestion par fournisseur.

9.9 Module ProfileManager

Ce module se compose de trois interfaces: ProfileConsumer, ProfileUsageMgr et ProfileRetriever. L'interface ProfileConsumer spécifie le contenu des notifications d'événement lors de créations de profil dans le répertoire d'objets de profil. L'interface ProfileUsageMgr prend en charge les opérations permettant de gérer les réglages de profil cachés dans le système de gestion par fournisseur. L'interface ProfileRetriever permet au système de gestion par fournisseur d'obtenir les valeurs pour les réglages de profil.

9.9.1 Interface ProfileConsumer

Le rôle de cette interface est d'annoncer l'existence de nouveaux réglages de profil et d'acheminer ces réglages jusqu'au système de gestion par fournisseur. Cette interface ne comporte pas d'opérations. Cependant elle fournit bien le mappage d'en-tête fixe ainsi que les mappages de données filtrables pour l'objet d'événement structuré servant à communiquer des informations relatives aux événements au moyen du canal d'événements du service de notification du groupe OMG.

Dans l'en-tête fixe, le type **domain_type** est mis à la valeur "télécommunications", le nom **type_name** est mis à la valeur "ProfileEvent" et le nom **event_name** est mis à la valeur d'une chaîne constante qui a la valeur "ProfileCreation" comme défini dans l'interface.

Le mappage dans les données filtrables suit une stratégie utilisant un identificateur de chaîne constante pour le composant de données filtrables suivi par la valeur de cet élément de données. De même, les éléments de données filtrables sont énumérés dans un ordre spécifique.

L'ordre des éléments filtrables est: ProfileName, ProfileType, EventTime, ProfileAttributeValues et NotificationIdentifier. La valeur de l'élément ProfileName a la syntaxe du type NameType. La valeur de l'élément ProfileType a la syntaxe "donnée courte non signée" et permet au consommateur (système de gestion par fournisseur) d'identifier le type de profil créé et de disperser les valeurs

d'attribut trouvées ultérieurement dans la structure de profil. La syntaxe des valeurs d'attribut de profil (ProfileAttributeValues) est profileStruct. L'élément EventTime a la syntaxe du type GeneralizedTimeType: c'est l'instant auquel le profil a été créé. L'élément NotificationIdentifier a la syntaxe du type NotificationIdentifierType. Ce identificateur donne une étiquette unique à l'événement de création de profil et est incrémenté avec chaque création de profil.

9.9.2 Interface ProfileUsageMgr

9.9.2.1 reName

Cette opération offre la capacité de renommer un profil.

La signature de l'opération **reName** est indiquée ci-dessous:

```
void reName (
            in NameType  oldProfileName,
            in NameType  newProfileName)
            raises  (UnknownProfiles, AccessDenied, DuplicateProfileName);
```

Le paramètre d'entrée **oldProfileName** est l'ancien nom de profil à renommer. Le paramètre d'entrée **newProfileName** est le nouveau nom pour le profil.

La valeur de retour est de type **void**.

9.9.2.2 inUse

Cette opération renvoie une valeur booléenne afin d'indiquer si le profil est en usage.

La signature de l'opération **inUse** est indiquée ci-dessous:

```
boolean inUse (
            in NameType profileName)
            raises (UnknownProfiles, AccessDenied);
```

Le paramètre d'entrée **ProfileName** donne le nom de profil qui doit être vérifié s'il est utilisé par un autre correspondant.

La valeur de retour est de type **booléen**.

9.9.2.3 suspendUse

Cette opération suspend l'utilisation d'un profil.

La signature de l'opération **suspendUse** est indiquée ci-dessous:

```
void suspendUse (
              in NameType profileName)
              raises (UnknownProfiles, AccessDenied);
```

Le paramètre d'entrée **profileName** est un nom de profil qui doit être suspendu.

La valeur de retour est de type **void**.

9.9.2.4 resumeUse

Cette opération reprend l'utilisation d'une profil nommé.

La signature de l'opération **resumeUse** est indiquée ci-dessous:

```
void resumeUse (
         in NameType profileName)
         raises (UnknownProfiles, AccessDenied) ;
```

Le paramètre d'entrée **profileName** nomme le profil dont les valeurs doivent être remises à disposition pour utilisation par le système de gestion du fournisseur.

La valeur de retour est de type **void**.

9.9.2.5 deleteProfile

Cette opération offre la capacité de supprimer un profil qui n'est pas en usage.

La signature de l'opération **deleteProfile** est indiquée ci-dessous:

```
void deleteProfile (in NameType profileName) raises (UnknownProfiles,
AccessDenied, ProfileInUse);
```

Le paramètre d'entrée **profileName** est le nom de profil à supprimer.

La valeur de retour est de type **void**.

9.9.2.6 Exceptions

L'exception **AccessDenied** est déclenchée quand le système OMS n'est pas autorisé à accéder à l'opération requise.

L'exception **DuplicateProfileName** est déclenchée quand un nom en double de profil est trouvé.

L'exception **ProfileInUse** est déclenchée lors de la suppression d'un profil et lorsque le profil est en usage par l'autre correspondant.

L'exception **UnknownProfiles** est déclenchée si le nom de profil fourni est inconnu du système de gestion par fournisseur et ne peut pas être extrait du répertoire d'objets de profil.

9.9.3 Interface ProfileRetriever

9.9.3.1 retrieve

Cette opération offre la capacité d'extraire un profil. Cette opération est utilisée par le système de gestion du fournisseur afin d'extraire un profil du système de gestion par opérateur.

La signature de l'opération **retrieve** est indiquée ci-dessous:

```
ProfileInfoType retrieve (
              in NameType profileName)
              raises (UnknownProfiles);
```

Le paramètre d'entrée **profileName** est le nom du profil à extraire.

La valeur de retour est de type **ProfileInfoType** et indique les valeurs d'attribut pour le profil nommé.

9.9.3.2 Exceptions

L'exception **UnknownProfiles** est déclenchée si le nom de profil fourni est inconnu du système de gestion par fournisseur et ne peut pas être extrait du répertoire d'objets de profil.

9.10 Module Registrar

Ce module prend en charge le processus de transfert d'un nouvel élément de réseau dans le domaine de gestion du système de gestion par fournisseur. Il prend en charge à la fois les installations initiales et les remplacements d'équipement motivés par la maintenance ou par des relèvements de service. Il prend explicitement en charge la fonction de télémétrie qui sert à mesurer le temps de propagation aller-retour entre la terminaison OLT et chaque terminaison ONU ou ONT et qui sert à établir des mécanismes de sécurité (algorithme à clé de brassage) et la temporisation des communications y compris l'installation automatique du canal d'exploitation intégré.

Un ensemble d'entités gérées est automatiquement créé dans le modèle d'informations de gestion conservé par le système de gestion du fournisseur à la suite d'une inscription. Selon le type d'équipement, des instances de terminaison OLT, ONT, ONU, APONLink, APONTTP,

APONTrail, APONNetworkCTP, APONNetworkTTP, APONLinkConnection, tcAdapterF, vpLinkConnectionF, vpTopologicalLinkF et physicalPathTPF sont automatiquement créées. Selon l'implémentation fournie ainsi que selon ce qui est physiquement installé, d'autres entités gérées sont également créées automatiquement.

9.10.1 Interface NERegistrar

9.10.1.1 registerNE

Cette opération inscrit un élément de réseau auprès d'une instance particulière du système de gestion par fournisseur. L'élément de réseau devrait être installé à ce point. L'identificateur d'entité gérée de l'élément de réseau est renvoyé afin d'indiquer que le système de gestion par fournisseur peut exécuter au moins une fonction de gestion rudimentaire.

La signature de l'opération **registerNE** est indiquée ci-dessous:

```
ManagedEntityIdType   registerNE (
                    in DCNAddressType nEDCNAddress,
                    in UserLabelType nEUserLabel,
                    in AdministrationDomainType administrationDomain)
                    raises (AccessDenied, DCNTimeout, AddressLabelMismatch,
                    DuplicateUserLabel, TooManyNEs, InvalidDCNAddress,
                    DeniedAccess, InvalidUserLabelSyntax);
```

Le paramètre d'entrée **nEDCNAddress** désigne une adresse RCD à utiliser afin d'accéder à cet élément de réseau, lequel devrait avoir déjà été configuré avec cette adresse. Le paramètre d'entrée **nEUserLabel** donne une étiquette définie par l'opérateur pour l'élément de réseau, lequel devrait avoir déjà été préfourni avec l'étiquette d'utilisateur. Le paramètre d'entrée **administrationDomain** désigne le domaine de gestion auquel cet élément de réseau est attribué.

La valeur de retour de type **ManagedEntityIdType** désigne l'élément de réseau nouvellement inscrit si cette opération est effectuée avec succès.

9.10.1.2 modifyNEDCNAddress

Cette opération modifie l'adresse du réseau de communication de données (RCD) d'un élément de réseau inscrit. Les communications peuvent être temporairement perdues entre le système de gestion par fournisseur et l'élément de réseau pendant cette opération. La nouvelle adresse RCD devrait être activée immédiatement.

La signature de l'opération **modifyNEDCNAddress** est indiquée ci-dessous:

```
void modifyNEDCNAddress (
       in ManagedEntityIdType    nEManagedEntityId,
       in DCNAddressType newNEDCNAddress)
       raises(AccessDenied, DeniedAccess, AddressLabelMismatch,
       DCNTimeout, CommFailure, UnknownNE, InvalidDCNAddress, BackupInProgress);
```

Le paramètre d'entrée **nEManagedEntityId** désigne l'élément de réseau auquel cette opération s'applique. Le paramètre d'entrée **newDCNAddress** désigne la nouvelle adresse RCD pour l'élément de réseau.

La valeur de retour est de type **void**.

9.10.1.3 rangeONTorONU

Cette opération télémesure une terminaison ONT ou ONU au moyen du numéro de série de l'interface PON de l'élément de réseau. Si l'élément de réseau en cours de télémétrie n'a pas été construit, alors l'opération de construction buildNode sera appelée avant que la télémétrie soit effectuée. Une fois la télémétrie effectuée avec succès, une synchronisation d'élément de réseau est lancée. Cela mettra à jour le système de gestion par fournisseur avec les données de configuration de l'élément de réseau nouvellement télémesuré.

La signature de l'opération **rangeONTorONU** est indiquée ci-dessous:

```
ManagedEntityIdType rangeONTorONU(
          in ManagedEntityIdType oLTManagedEntityId,
          in UserLabelType nEUserLabel,
          in SerialNumType serialNum,
          in ManagedEntityIdType port)
          raises(AccessDenied, CommFailure, EquipmentFailure,
          UnknownNE, UnknownPort, MaxSubtendingNodesExceeded, InsufficientPONBW,
          InvalidSerialNumSyntax, APONLayerFailure, DuplicateUserLabel,
          InvalidUserLabelSyntax, BackupInProgress, SynchInProgress );
```

Le paramètre d'entrée **oLTManagedEntityId** désigne la terminaison OLT à laquelle cette opération s'applique. Le paramètre d'entrée **nEUserLabel** est une étiquette attribuée à l'élément de réseau à télémesurer. Le paramètre d'entrée **serialNum** désigne le numéro de série de l'élément de réseau. Le paramètre d'entrée **port** désigne l'accès spécifique de réseau PON sur la terminaison OLT où l'élément de réseau est à télémesurer.

La valeur de retour de type **ManagedEntityIdType** est l'identificateur d'entité gérée de l'élément de réseau télémesuré.

9.10.1.4 rangeReplacementNE

Cette opération télémesure une terminaison de remplacement ONT ou ONU au moyen du numéro de série de l'équipement de remplacement. Toutes les informations de connexion de service existantes sont automatiquement attribuées à l'élément de réseau de remplacement. Des scénarios de remplacement se produisent en raison des activités de maintenance pour des clients et services existants. L'ancienne et la nouvelle étiquette d'utilisateur d'élément de réseau peuvent être identiques. En dehors du domaine d'application de l'interface IF1, le fournisseur et l'opérateur sont parvenus à une entente sur la question de savoir quel type de matériel peut remplacer un matériel existant.

La signature de l'opération **rangeReplacementNE** est indiquée ci-dessous:

```
ManagedEntityIdType rangeReplacementNE(
               in ManagedEntityIdType oldNEManagedEntityId,
               in UserLabelType newNEUserLabel,
               in SerialNumType replacementSerialNum)
               raises (AccessDenied, CommFailure, UnknownNE,
               InvalidSerialNumSyntax, APONLayerFailure, EquipmentFailure,
               InvalidUserLabelSyntax, HWServicesMismatch,
               DuplicateUserLabel, BackupInProgress, SynchInProgress );
```

Le paramètre d'entrée **oldNEManagedEntityId** désigne l'élément de réseau à remplacer. Le paramètre d'entrée **newNEUserLabel** désigne une étiquette pour la nouvelle terminaison ONT ou ONU à télémesurer. Le paramètre d'entrée **replacementSerialNum** désigne le numéro de série de l'élément de réseau.

La valeur de retour de type **ManagedEntityIdType** est l'identificateur d'entité gérée de l'élément de réseau nouvellement télémesuré.

9.10.1.5 rangeUpgradeNE

Cette opération télémesure un élément de réseau de remplacement afin de mettre à jour le matériel. Toutes les informations de connexion de service existantes sont automatiquement attribuées à l'élément de réseau de remplacement. Par ailleurs, l'on part du principe que l'élément de réseau de remplacement a été préfourni (au moyen de l'opération buildNode) et que les éventuelles connexions de service ont été préfournies ou que de la largeur de bande a été réservée. La nouvelle étiquette d'utilisateur d'élément de réseau peut être la même que l'ancienne. En dehors du domaine

d'application de l'interface IF1, le fournisseur et l'opérateur sont parvenus à une entente sur la question de savoir quel type de matériel peut remplacer un matériel existant.

La signature de l'opération **rangeUpgradeNE** est indiquée ci-dessous:

```
ManagedEntityIdType rangeUpgradeNE(
               in ManagedEntityIdType oldNEManagedEntityId,
               in ManagedEntityIdType newNEManagedEntityId,
               in UserLabelType newNEUserLabel,
               in SerialNumType newNESerialNum )
               raises (AccessDenied, CommFailure,
               APONLayerFailure, EquipmentFailure, InvalidUserLabelSyntax,
               DuplicateUserLabel, UnknownNE,
               HWServicesMismatch, InsufficientPONBW, BackupInProgress,
               SynchInProgress );
```

Le paramètre d'entrée **oldNEManagedEntityId** désigne l'élément de réseau qui est remplacé. Dans ce cas, le type d'élément de réseau qui est remplacé est soit une terminaison ONT ou une terminaison ONU. Le paramètre d'entrée **newNEManagedEntityId** désigne le remplacement préfourni. Toutes les informations relatives aux nouvelles connexions de service ou à la réservation de largeur de bande sont accessibles au système de gestion par fournisseur au moyen de cet identificateur. Le paramètre d'entrée **newNEUserLabel** donne une étiquette au nouvel élément de réseau à télémesurer. L'ancienne et la nouvelle étiquette d'utilisateur de terminaison ONU ou ONT peuvent être identiques. Le paramètre d'entrée **newNESerialNum** donne le numéro de série à utiliser afin de télémesurer l'élément de remplacement.

La valeur de retour de type **ManagedEntityIdType** est l'identificateur d'entité gérée de l'élément de réseau nouvellement télémesuré.

9.10.1.6 moveONTorONU

Cette opération sert à déplacer une terminaison ONT ou ONU d'un réseau PON à un autre et également de déplacer tous les services associés.

La signature de l'opération **moveONTorONU** est indiquée ci-dessous:

```
ManagedEntityIdType moveONTorONU(
                    in ManagedEntityIdType oldNEManagedEntityId,
                    in ManagedEntityIdType newPONPort)
                    raises(AccessDenied, CommFailure, UnknownNE, UnknownPort,
                    APONLayerFailure,EquipmentFailure, InsufficientPONBW,
                    BackupInProgress, SynchInProgress );
```

Le paramètre d'entrée **oldNEManagedEntityId** désigne l'élément de réseau à remplacer. Le paramètre d'entrée **newPONPort** désigne le nouveau réseau PON auquel ces services sont transférés.

La valeur de retour de type **ManagedEntityIdType** est l'identificateur d'entité gérée de l'élément de réseau nouvellement télémesuré.

9.10.1.7 getSubtendingNEList

Cette opération renvoie tous les sous-éléments d'un élément de réseau particulier.

La signature de l'opération **getSubtendingNEList** est indiquée ci-dessous:

```
ManagedEntityIdSeqType getSubtendingNEList(
                    in ManagedEntityIdType nEManagedEntityId)
                    raises (UnknownNE, AccessDenied);
```

Le paramètre d'entrée **nEManagedEntityId** désigne l'élément de réseau spécifique auquel la liste devrait être renvoyée.

La valeur de retour de type **ManagedEntityIdSeqType** donne une liste de sous-éléments de réseau.

9.10.1.8 nEListGet

Cette opération extrait les éléments de réseau se trouvant dans le domaine de gestion du système de gestion par fournisseur.

La signature de l'opération **nEListGet** est indiquée ci-dessous:

```
ManagedEntityIdSeqType nEListGet () raises (AccessDenied);
```

Il n'y a aucun paramètre d'entrée.

La valeur de retour est de type **ManagedEntityIdSeqType** qui indique la liste des éléments de réseau gérés par le système de gestion du fournisseur.

9.10.1.9 deRegisterNE

Cette opération supprime l'élément de réseau du domaine de gestion du système de gestion par fournisseur.

La signature de l'opération **deRegisterNE** est indiquée ci-dessous:

```
void deRegisterNE (
            in ManagedEntityIdType nE)
            raises (AccessDenied);
```

Le paramètre d'entrée **nE** identifie l'élément de réseau à supprimer du domaine de gestion du système de gestion par fournisseur.

La valeur de retour est de type **void.**

9.10.1.10 associateNE

Cette opération associe les informations de préfourniture à un élément de réseau qui a été installé et autodécouvert. Les activités de préfourniture et d'installation produisent l'une et l'autre une entité gérée dans le système de gestion par fournisseur qui est mise à la disposition du système OMS. Cette opération unifie les deux entités gérées.

La signature de l'opération **associateNE** est indiquée ci-dessous:

```
ManagedEntityIdType associateNE (
                  in ManagedEntityIdType preProvisionedNE,
                  in ManagedEntityIdType discoveredNE)
                  raises (AccessDenied, UnknownManagedEntity);
```

Le paramètre d'entrée **preProvisionedNE** identifie les informations associées à l'élément de réseau préfourni. Le paramètre d'entrée **discoveredNE** identifie les informations associées à l'élément de réseau installé.

La valeur de retour de type **ManagedEntityIdType** donne l'identification des informations fusionnées.

9.10.1.11 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **AddressLabelMismatch** est déclenchée quand l'élément de réseau désigné ne possède pas l'adresse RCD actuellement indiquée dans la demande.

L'exception **APONLayerFailure** est déclenchée quand il y a eu une panne de télémétrie par protocole de réseau APON entre la terminaison OLT et le nœud sous-jacent désigné.

L'exception **BackupInProgress** est déclenchée quand la demande d'annulation est émise pendant que la sauvegarde est en cours.

L'exception **CommFailure** est déclenchée en cas de panne du RCD entre système de gestion par fournisseur et terminaison OLT ou de panne de communication entre terminaison OLT et terminaison ONT de départ.

L'exception **DCNTimeout** est déclenchée quand la liaison de communications par RCD entre au moins un des éléments de réseau et le système de gestion par fournisseur est encombrée au point que les informations d'état actuel ou de statut ne peuvent pas être transférées dans un intervalle de synchronisation défini par le système.

L'exception **DeniedAccess** est déclenchée si l'accès à l'élément de réseau est refusé en conséquence de restrictions relatives au contrôle d'accès.

L'exception **DuplicateUserLabel** est déclenchée si l'étiquette d'utilisateur fournie dans la demande a servi à étiqueter un autre élément de réseau ou module d'extension. En d'autres termes, le système de gestion par fournisseur est chargé de surveiller les étiquettes d'utilisateur attribuées aux éléments de réseau et aux modules d'extension dans son domaine de gestion.

L'exception **EquipmentFailure** est déclenchée quand l'équipement où les données sont sauvegardées se trouve en état de panne.

L'exception **HWServicesMismatch** est déclenchée quand l'élément de réseau de remplacement ne peut pas exécuter les services préfournis.

L'exception **InsufficientPONBW** est déclenchée quand la terminaison ONT ou ONU ne peut pas être télémesurée en raison d'une largeur de bande insuffisante sur la liaison de réseau APON.

L'exception **InvalidDCNAddress** est déclenchée quand l'adresse RCD spécifiée n'est pas valide.

L'exception **InvalidSerialNumSyntax** est déclenchée quand la syntaxe du numéro de série fourni ne concorde pas avec les règles de définition.

L'exception **InvalidUserLabelSyntax** est déclenchée quand l'étiquette d'utilisateur fournie pour la terminaison ONU ou ONT viole les règles de syntaxe commerciales définies par l'opérateur et implémentées dans le système de gestion par fournisseur.

L'exception **MaxSubtendingNodesExceeded** est déclenchée quand le nombre maximal configuré de nœuds sous-jacents pour l'interface PON désignée a été dépassé par cette demande de fourniture de services.

L'exception **SynchInProgress** est déclenchée quand une opération est requise pendant que le système de gestion par fournisseur est en cours de synchronisation avec l'élément de réseau.

L'exception **TooManyNEs** est déclenchée quand le système de gestion par fournisseur ne peut pas gérer une terminaison OLT supplémentaire.

L'exception **UnknownManagedEntity** est déclenchée quand l'équipement est inconnu du système de gestion par fournisseur.

L'exception **UnknownNE** est déclenchée quand la terminaison OLT est inconnue du système de gestion par fournisseur.

L'exception **UnknownPort** est déclenchée quand l'accès désigné est inconnu du système de gestion par fournisseur.

9.11 Module ResourceAllocation

Ce service permet au système OMS de réserver de la largeur de bande dans des ressources de système pour une connexion de service anticipée. La largeur de bande réservée peut être supprimée ou extraite. L'interface ResourceAllocator permet de créer, de supprimer ou d'afficher une

réservation de ressource. Cette opération est utilisée avant l'envoi de personnel pour l'installation d'un élément de réseau. Une fois que la capacité d'une ressource est réservée, la ressource réservée ne peut être utilisée que pour le service spécifié dans la réservation.

9.11.1 Interface ResourceAllocator

9.11.1.1 reserveForService

Cette opération réserve de la largeur de bande pour une ressource réseau telle qu'une terminaison ONT, ONU ou NT dont l'installation est en instance. Cette opération est utilisée quand la terminaison ONT, ONU ou NT qui dessert la largeur de bande associée à la terminaison OLT désignée est préfournie pour la première fois. Quand l'opération est exécutée, la valeur de retour, **ReservationBandwidthType**, fournit au système OMS un compte rendu de la largeur de bande réservée.

La signature de l'opération **reserveForService** est indiquée ci-dessous:

```
ReservationBandwidthType reserveForService(
                        in EndPointType endPointA,
                        in EndPointType endPointZ,
                        in NameSeqType networkCharacteristicsProfiles,
                        in ServiceInstanceIdType serviceInstanceId)
                        raises(UnknownNE, UnknownPort, UnknownProfiles,
                        InsufficientBW, MaxSubtendingNodesExceeded,
                        ConnectionCountExceeded, CommFailure, AccessDenied,
                        ProfileSuspended );
```

Les paramètres d'entrée **endPointA** et **endPointZ** identifient les deux extrémités de la connexion de service qui détermineront les ressources nécessaires afin de prendre en charge une connexion de service anticipée. Le paramètre d'entrée **networkCharacteristicsProfiles** est une liste composée des éléments suivants: abTrafficDescripterProfile et baTrafficDescripterProfile (dans cet ordre). Le paramètre d'entrée **serviceInstanceId** désigne l'instance de service associée à cette largeur de bande réservée.

La valeur de retour est de type **ReservationBandwidthType** et comprend un identificateur de réservation et la quantité de ressource réseau en terme de largeur de bande qui a été réservée par le système de gestion du fournisseur. L'identificateur de réservation peut être utilisé par le système OMS pendant la fourniture de services ou afin d'annuler cette réservation.

9.11.1.2 cancelReservation

Cette opération sert à supprimer la réservation et libérer les ressources de la capacité réservée du système.

La signature de l'opération **cancelReservation** est indiquée ci-dessous:

```
AvailableSysBandwidthSeqType cancelReservation (
                             in ReservationIdType reservationId )
                             raises (UnknownReservationId, CommFailure,
                             AccessDenied);
```

Le paramètre d'entrée **reservationId** désigne la réservation actuellement associée à la largeur de bande qui a été attribuée.

La valeur de retour de type **AvailableSysBandwidthSeqType** indique la largeur de bande actuellement disponible de la terminaison OLT après suppression de la largeur de bande réservée.

9.11.1.3 getReservationId

Cette opération sert à afficher l'identificateur de réservation associé à l'identificateur d'instance de service qui est attribué à la largeur de bande réservée.

La signature de l'opération **getReservationId** est indiquée ci-dessous:

```
ReservationIdType getReservationId (
                 in ServiceInstanceIdType serviceInstanceId)
                 raises(UnknownServiceInstance, AccessDenied);
```

Le paramètre d'entrée **serviceInstanceId** sert à identifier les instances de service actuellement attribuées pendant que la ressource est réservée.

La valeur de retour de type **ReservationIdType** est associée à l'identificateur d'instance de service indiqué.

9.11.1.4 reportReservedResources

Cette opération est utilisée par le système OMS afin d'afficher la largeur de bande actuellement réservée dans l'élément NE spécifique de tête de réseau (dans ce cas la terminaison OLT).

La signature de l'opération **reportReservedResources** est indiquée ci-dessous:

```
ReservedBandwidthSeqType reportReservedResources (
                            in ManagedEntityIdType nEManagedEntityId)
                            raises (UnknownNE, AccessDenied);
```

Le paramètre d'entrée **nEManagedEntityId** sert à identifier l'élément de réseau pour lequel les réservations actuelles de largeur de bande sont demandées.

La valeur de retour est de type **ReservedBandwidthSeqType** et possède toutes les informations de largeur de bande information pour l'élément de réseau.

9.11.1.5 getReservations

Cette opération est utilisée par le système OMS afin d'extraire toutes les réservations associées à l'élément de réseau considéré.

La signature de l'opération **getReservations** est indiquée ci-dessous:

```
ReservationIdSeqType getReservations(
                            in ManagedEntityIdType nEManagedEntityId)
                            raises (UnknownNE, AccessDenied);
```

Le paramètre d'entrée **nEManagedEntityId** sert à identifier l'élément de réseau pour extraction de la largeur de bande actuellement réservée.

La valeur de retour de type **ReservationIdSeqType** donne tous les identificateurs de réservation actuellement attribués à la terminaison OLT.

9.11.1.6 cancelAllRemainingReservations

Cette opération sert à supprimer toutes les réservations restantes en fonction de la capacité associée à un élément de réseau donné. Celui-ci peut avoir de la largeur de bande qui est soit attribuée, réservée ou disponible pour des connexions de service. Cette opération change seulement des ressources réservées en ressources disponibles.

La signature de l'opération **cancelAllRemainingReservations** est indiquée ci-dessous:

```
AvailableSysBandwidthSeqType cancelAllRemainingReservations(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, CommFailure, AccessDenied);
```

Le paramètre d'entrée **nEManagedEntityId** sert à identifier l'élément de réseau.

La valeur de retour de type **AvailableSysBandwidthSeqType** indique la largeur actuellement disponible de l'élément de réseau après la suppression de la largeur de bande réservée.

9.11.1.7 getReservation

Cette opération sert à rechercher les origines d'une réservation de capacité dans un élément de réseau.

La signature de l'opération **getReservation** est indiquée ci-dessous:

```
ReservationInfoType getReservation (
                  in ReservationIdType reservationId)
                  raises (UnknownReservationId, AccessDenied);
```

Le paramètre d'entrée **reservationId** sert à spécifier la réservation.

La valeur de retour est de type **ReservationIdInfoType** indique les informations de connexion de service fournies dans le cadre de la demande de réservation avec l'identificateur **reservationId** fourni.

9.11.1.8 getAvailableSysBandwidth

Cette opération sert à déterminer la grandeur de capacité restant pour réservation ou attribution de connexions de service pour un élément de réseau.

La signature de l'opération **getAvailableSysBandwidth** est indiquée ci-dessous:

```
AvailableSysBandwidthSeqType getAvailableSysBandwidth (
                   in ManagedEntityIdType nEManagedEntityId)
                   raises (UnknownNE, CommFailure, AccessDenied);
```

Le paramètre d'entrée **nEManagedEntityId** sert à identifier l'élément de réseau.

La valeur de retour de type **AvailableSysBandwidthSeqType** indique la largeur actuellement disponible de l'élément de réseau. Elle indique la largeur de bande disponible à chaque accès préfourni sur l'élément de réseau, comme caractérisé par l'implémentation fournie.

9.11.1.9 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **CommFailure** est déclenchée en cas de panne du RCD entre système de gestion par fournisseur et terminaison OLT ou en cas de panne de communication entre terminaisons OLT et ONT ou ONU.

L'exception **ConnectionCountExceeded** est déclenchée quand le nombre maximal de connexions pour la terminaison OLT ou l'accès de réseau PON a été dépassé par cette demande de fourniture de services.

L'exception **InsufficientBW** est déclenchée quand la largeur de bande ne suffit pas au service requis.

L'exception **MaxSubtendingNodesExceeded** est déclenchée quand le nombre maximal configuré de nœuds sous-jacents pour l'interface PON désignée a été dépassé par cette demande de fourniture de services.

L'exception **ProfileSuspended** est déclenchée quand les profils nommés dans l'invocation ont été suspendus pour utilisation par le système OMS ou par l'opérateur dans le système de gestion par fournisseur.

L'exception **UnknownNE** est déclenchée quand la terminaison OLT est inconnue du système de gestion par fournisseur.

L'exception **UnknownPort** est déclenchée quand l'accès désigné est inconnu du système de gestion par fournisseur.

L'exception **UnknownProfiles** est déclenchée si le nom de profil fourni est inconnu du système de gestion par fournisseur et ne peut pas être extrait du répertoire d'objets de profil.

L'exception **UnknownReservationId** est déclenchée quand le système de gestion par fournisseur ne reconnaît pas cet identificateur de réservation.

L'exception **UnknownServiceInstance** est déclenchée quand l'instance de service est inconnue du système de gestion par fournisseur.

9.12 Module SchedulerManagement

Ce service sert à fournir des interfaces de système OMS afin de gérer les programmateurs de planification à utiliser afin d'invoquer diverses activités. Une fois que le programmateur est créé, le système OMS peut lancer des demandes de planification d'activités telles que la téléexportation vers la base MIB d'un élément de réseau, le transfert en masse, les essais ou la téléimportation de logiciels par référence au programmateur de planification. Les programmateurs de planification sont déterminés par les besoins de l'environnement d'exploitation. L'on part du principe qu'il n'y aura aucun programmateur de planification nommé ou établi automatiquement sur instanciation du système de gestion par fournisseur. Les références à un programmateur spécifique seront toujours effectuées au moyen d'étiquettes d'utilisateur. Une matrice de déclencheurs sert à décrire le programme de planification. Les valeurs contenues dans la matrice sont interprétées d'après la valeur de l'indicateur HourlyDailyWeeklyMonthlyInd.

9.12.1 Interface SchedulerMgr

9.12.1.1 makeScheduler

Cette opération crée un nouvel objet de programmateur. Le système OMS peut alors associer des activités à l'objet de programmateur par référence au nom de l'étiquette d'utilisateur du programmateur.

La signature de l'opération **makeScheduler** est indiquée ci-dessous:

```
void makeScheduler (
            in UserLabelType schedulerName,
            in GeneralizedTimeType startTime,
            in GeneralizedTimeType stopTime,
            in HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd,
            in TriggerTimeMatrixSeqType matrix)
            raises (InvalidStartTime, InvalidStopTime, DuplicateUserLabel,
            MatrixSchedulerTypeMismatch, AccessDenied, InvalidTrigger);
```

Le paramètre d'entrée **schedulerName** désigne le programmateur qui peut être cité en référence par différentes activités. Les paramètres d'entrée **startTime** et **stopTime** identifient l'intervalle temporel dans lequel l'activité ou les activités planifiées sont applicables. Le paramètre d'entrée **hourlyDailyWeeklyMonthlyInd** indique la fréquence (toutes les heures, tous les jours, toutes les semaines ou tous les mois) à laquelle le programme de planification devrait être déclenché. Le paramètre d'entrée **matrix** donne des informations spécifiques d'invocation de planification et est affecté par la valeur du paramètre **hourlyDailyWeeklyMonthlyInd**.

Le Tableau 4 ci-dessous donne un tableau montrant la relation entre les paramètres **dailyWeeklyMonthlyInd** et **matrix**:

Tableau 4/Q.834.4 – Détails de la matrice

Valeur de hourlyDailyWeeklyMonthlyInd	Valeur du paramètre matrix
Toutes les heures	Instant: toute valeur comprise entre 0 et 3 600[7] dayOfWeek: doit toujours être 'non spécifié' dayOfMonth: doit toujours être 0
Tous les jours	Instant: toute valeur comprise entre 0 et 86 400[8] dayOfWeek: doit toujours être 'non spécifié' dayOfMonth: doit toujours être 0
Toutes les semaines	Instant: toute valeur comprise entre 0 et 86 400 dayOfWeek: différent de 'non spécifié' dayOfMonth: doit toujours être 0
Tous les mois	Instant: toute valeur comprise entre 0 et 86 400 dayOfWeek – doit toujours être 'non spécifié' dayOfMonth – différent de 0

La valeur de retour est de type **void**.

9.12.1.2 suspendScheduler

Cette opération sert à suspendre un programme de planification. Pour l'essentiel, elle règle la valeur de l'état administratif du programme de planification entre 'unlocked' et 'locked'. Elle sera applicable à partir de la prochaine itération de planification. Cette opération provoque la suspension de toute activité de planification associée.

La signature de l'opération **suspendScheduler** est indiquée ci-dessous:

```
void suspendScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied);
```

Le paramètre d'entrée **schedulerName** sert à identifier le programme de planification à suspendre.

La valeur de retour est de type **void**.

9.12.1.3 resumeScheduler

Cette opération sert à reprendre un programme de planification suspendu. Pour l'essentiel, elle règle la valeur de l'état administratif du programme de planification de 'locked' à 'unlocked'. Elle sera applicable à partir de la prochaine itération de planification. Cette opération provoque la reprise de toute activité de planification associée.

La signature de l'opération **resumeScheduler** est indiquée ci-dessous:

```
void resumeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied );
```

Le paramètre d'entrée **schedulerName** sert à identifier le programme de planification à reprendre.

La valeur de retour est de type **void**.

---

[7] Représente le nombre de secondes à partir du début de l'heure contenant l'instant de déclenchement.

[8] Représente le nombre de secondes à partir du début du jour contenant l'instant de déclenchement, le jour commençant après minuit.

9.12.1.4 modifyTime

Cette opération sert à modifier l'instant de début **startTime** et l'instant de fin **stopTime** d'un programme de planification. Elle est utilisée par le système OMS afin d'allonger ou de raccourcir l'intervalle temporel dans lequel le programme de planification est applicable.

La signature de l'opération **modifyTime** est indiquée ci-dessous:

```
void modifyTime (
            in UserLabelType schedulerName,
            in GeneralizedTimeType newStartTime,
            in GeneralizedTimeType newStopTime)
            raises (InvalidStartTime, InvalidStopTime, UnknownScheduler,
            AccessDenied);
```

Le paramètre d'entrée **schedulerName** sert à identifier le programme de planification. Le paramètre d'entrée **newStartTime** indique l'instant auquel ce programmateur commence et le paramètre d'entrée **newStopTime** indique l'instant auquel le programmateur s'arrête. Si la valeur de l'élément newStartType est antérieure à l'instant actuel du système, l'exception InvalidStartTime est déclenchée. Une valeur de 0 pour soit **newStartTime** ou **newStopTime** indique que la valeur précédente ne devrait pas être modifiée.

La valeur de retour est de type **void**.

9.12.1.5 changeSchedulerName

Cette opération sert à changer le nom de l'objet de programmateur. En cas de succès de cette opération, le nouveau nom du programmateur sera immédiatement applicable. Toute référence à ce programme devra être faite au moyen du nom nouvellement attribué.

De nombreuses activités peuvent être associées au même nom de programmateur. Le fait de changer le nom du programmateur n'aura pas d'incidence sur les activités associées au programmateur. Toutes les activités associées au programmateur modifié conserveront leur association.

La signature de l'opération **changeSchedulerName** est indiquée ci-dessous:

```
void changeSchedulerName (
            in UserLabelType oldSchedulerName,
            in UserLabelType newSchedulerName)
            raises (UnknownScheduler, DuplicateUserLabel, AccessDenied);
```

Le paramètre d'entrée **oldSchedulerName** sert à identifier le programme de planification existant. Le paramètre d'entrée **newSchedulerName** sert à indiquer le nouveau nom à attribuer à ce programmateur.

La valeur de retour est de type **void**.

9.12.1.6 modifyTriggerTimes

Cette opération est utilisée par le système OMS afin de spécifier nouveaux instants de déclenchement et de nouvelles itérations pour un programmateur[9]. Si cette opération réussit, alors toute activité de planification associée se produira aux nouveaux instants de planification à partir de la prochaine itération de planification.

---

9 Voir l'opération makeScheduler pour les détails sur les variables de dépendance utilisées.

La signature de l'opération **modifyTriggerTimes** est indiquée ci-dessous:

```
void modifyTriggerTimes (
                  in UserLabelType schedulerName,
                  in HourlyDailyWeeklyMonthlyIndType
                  newHourlyDailyWeeklyMonthlyInd,
                  in TriggerTimeMatrixSeqType newMatrix)
                  raises (UnknownScheduler, MatrixSchedulerTypeMismatch,
                  InvalidTrigger, AccessDenied );
```

Le paramètre d'entrée **schedulerName** sert à identifier le programme de planification à modifier. Le paramètre d'entrée **newHourlyDailyWeeklyMonthlyInd** indique la nouvelle fréquence à laquelle le programme de planification devrait être déclenché. Le paramètre d'entrée **newMatrix** sert à fournir les nouvelles informations spécifiques d'invocation de planification. Voir le Tableau 4 ci-dessus concernant la relation entre les variables. Les deux paramètres **newDailyWeeklyMonthlyInd** et **newMatrix** doivent toujours être explicitement spécifiés.

La valeur de retour est de type **void**.

9.12.1.7 removeScheduler

Cette opération sert à supprimer un programmateur. Cette opération ne sera autorisée que s'il n'y a aucune activité associée à ce programmateur.

La signature de l'opération **removeScheduler** est indiquée ci-dessous:

```
void removeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied, ScheduleInUse );
```

Le paramètre d'entrée **schedulerName** sert à identifier le programme de planification à supprimer.

La valeur de retour est de type **void**.

9.12.1.8 retrieveScheduler

Cette opération sert à extraire des informations sur le programme de planification. L'objet de programme de planification sera renvoyé après invocation efficace de cette opération.

La signature de l'opération **retrieveScheduler** est indiquée ci-dessous:

```
SchedulerType retrieveScheduler (
                        in UserLabelType schedulerName)
                        raises (UnknownScheduler, AccessDenied);
```

Le paramètre d'entrée **schedulerName** sert à identifier le programme de planification à afficher.

La valeur de retour est de type **Scheduler** et indique les informations suivantes: schedulerName; startTime; stopTime; hourlyDailyWeeklyMonthlyInd; matrix; operationalState et administrativeState.

9.12.1.9 schedulerListGet

Cette opération sert à extraire les noms de tous les programmateurs de planification actuellement définis pour le système de gestion par fournisseur.

La signature de l'opération **schedulerListGet** est indiquée ci-dessous:

```
SchedulerSeqType schedulerListGet ()
                  raises (AccessDenied);
```

Il n'y a aucun paramètre d'entrée.

La valeur de retour est de type **SchedulerSeqType** et indique l'énumération recherchée.

9.12.1.10 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **DuplicateUserLabel** est déclenchée si l'étiquette d'utilisateur fournie dans la demande a servi à étiqueter un autre programmateur. En d'autres termes, le système de gestion par fournisseur est chargé de surveiller les étiquettes d'utilisateur attribuées aux programmateurs de planification dans son domaine de gestion.

L'exception **InvalidStartTime** est déclenchée quand l'instant de début spécifié est incompatible avec la matrice actuelle des déclencheurs temporels ou avec l'instant d'arrêt.

L'exception **InvalidStopTime** est déclenchée quand l'instant d'arrêt spécifié est incompatible avec la matrice actuelle des déclencheurs temporels ou avec l'instant de début.

L'exception **InvalidTrigger** est déclenchée quand le déclencheur spécifié a des valeurs qui ne peuvent pas être interprétées par le programmateur de planification.

L'exception **MatrixSchedulerTypeMismatch** est déclenchée quand la syntaxe de la matrice des déclencheurs temporels ne correspond pas au type de programmateur nommé.

L'exception **ScheduleInUse** sera renvoyée par l'opération **removeScheduler** s'il y a des opérations associées à la planification.

L'exception **UnknownScheduler** est déclenchée quand le nom indiqué du programmateur n'est pas trouvé.

9.13 Module ServiceProvisioning

Dans ce module, le système de gestion par fournisseur sélectionne des accès, des ressources et de la largeur de bande afin d'être en mesure d'effectuer le processus de conception, de sélection et d'attribution qui est associé à un service.

9.13.1 Interface ServiceProvisioner

9.13.1.1 provisionConnection

Cette opération préfournit une connexion entre deux extrémités quelconques d'un système d'accès par fibre optique à un réseau BPON. Une connexion peut être établie entre une interface NNI et une interface UNI, entre deux interfaces UNI ou entre deux interfaces NNI. La Figure 1 décrit ces extrémités.

La signature de l'opération **provisionConnection** est indiquée ci-dessous:

```
ManagedEntityIdType provisionConnection(
                    in EndpointType endPointA,
                    in EndpointType endPointZ,
                    in NameSeqType networkCharacteristicsProfiles,
                    in ServiceInstanceIdType serviceInstanceId,
                    in AdministrativeStateType  administrativeState)
                    raises (UnknownNE, UnknownProfiles, UnknownPort,
                    InsufficientBW, ConnectionCountExceeded, CommFailure,
                    EquipmentFailure, ParameterViolation,AccessDenied,
                    InsufficientPONBW, ProfileSuspended,
                    ConnectionAlreadyExists);
```

Les paramètres d'entrée **endPointA** et **endPointZ** désignent les deux extrémités de la connexion demandée. Une extrémité est définie par la structure de données indiquée dans le Tableau 5.

Tableau 5/Q.834.4 – Détails des extrémités

Nom du champ	Définition	Syntaxe	Commentaires
portId	Spécifie l'accès physique contenant une extrémité de la connexion.	ManagedEntityIdType	L'on suppose qu'il s'agit de l'entité gérée physicalPathTP pour l'accès
endPointParameters	Désigne des paramètres spécifiques d'instance de service facilitant le regroupement de composants de connexion réseau.	any	Structures à fournir dans le cadre de l'implémentation. Une connexion ATM par exemple aurait les paramètres VPI, VCI.
serviceCharacteristics Profile	Enumère les références aux profils caractérisant le service à l'extrémité.	NameSeqType	Exemples: AAL1Profile, AAL5Profile, ATMNetworkAccess Profile, UNIProfile, CESServiceProfi le, DS1Profile, DS3Profile, EthernetProfile, AAL2Profile, LESProfile, SSCSParameter Profile1, SSCSParameter Profile2, VoiceServiceProfile AAL2, BridgedLANService Profile, MACBridgeService Profile. (Note)
NOTE – Les choix spécifiques dépendent des caractéristiques des services et des équipements. Les profils sont énumérés selon les noms fournis dans le module Q834ProfileManager. Voir des exemples de relations dans l'Annexe D.			

Le paramètre d'entrée **networkCharacteristicsProfiles** donne la liste des profils relatifs au transport à utiliser pour la fourniture de services, y compris les profils azTrafficDescriptorProfile et zaTrafficDescriptorProfile. Les profils de type descripteur de trafic *az* sont cités avant les profils de type *za* dans l'énumération. Le paramètre **serviceInstanceId** indique l'identificateur d'instance de service à utiliser comme clé lors d'une référence à des ressources de réseau associées au service. Le paramètre d'entrée **administrativeState** spécifie si la connexion de sous-réseau est en mesure d'écouler le trafic d'abonnés une fois que cette opération est exécutée.

La valeur de retour de type **ManagedEntityIdType** est un identificateur d'entité gérée qui permet d'identifier la connexion de sous-réseau créée à la suite de cette demande.

9.13.1.2 provisionReservation

Cette opération préfournit un service entre l'interface NNI d'une terminaison OLT et l'interface UNI d'une terminaison ONT ou entre deux interfaces UNI sur la base d'une réservation en instance. Une

fois que la connexion de service a été préfournie, l'identificateur de largeur de bande réservée et de la réservation associée est retiré du système de gestion par fournisseur parce que les ressources réservées sont attribuées.

La signature de l'opération **provisionReservation** est indiquée ci-dessous:

```
ManagedEntityIdType provisionReservation(
                  in ReservationIdType reservationId,
                  in AdministrativeStateType  administrativeState)
                  raises ( UnknownReservationId, AccessDenied);
```

Le paramètre d'entrée **reservationId** spécifie l'identificateur de réservation. Il désigne toutes les autres informations (telles que l'identificateur d'instance de service et les extrémités) à associer à la connexion préfournie. Le paramètre d'entrée **administrativeState** spécifie si la connexion de sous-réseau est en mesure d'écouler le trafic d'abonnés une fois que cette opération est exécutée.

La valeur de retour de type **ManagedEntityIdType** est un identificateur d'entité gérée permettant d'identifier la connexion de sous-réseau créée à la suite de cette demande.

9.13.1.3 deleteConnection

Cette opération détruit le service et les connexions qui existent.

La signature de l'opération **deleteConnection** est indiquée ci-dessous:

```
void deleteConnection(
            in ManagedEntityIdType subnetworkConnectionId)
            raises (UnknownConnection, CommFailure, EquipmentFailure,
            AccessDenied);
```

Le paramètre d'entrée **subnetworkConnectionId** est le sous-réseau qui a été créé antérieurement au moyen d'une demande de fourniture de services.

La valeur de retour est de type **void**.

9.13.1.4 modifyConnection

Cette opération offre la capacité de modifier le service existant.

La signature de l'opération **modifyConnection** est indiquée ci-dessous:

```
ManagedEntityIdType  modifyConnection (
                  in ManagedEntityIdType subnetworkConnectionId,
                  in ManagedEntityIdType  portB,
                  in NameSeqType newNetworkCharacteristicsProfiles,
                  in NameSeqType newServiceCharacteristicsProfiles)
                  raises(UnknownConnection, UnknownProfiles, InsufficientBW,
                  UnknownPort, AccessDenied, ProfileSuspended );
```

Le paramètre d'entrée **subnetworkConnectionId** est le sous-réseau qui a été créé antérieurement au moyen d'une demande de fourniture de services. Le paramètre d'entrée **portB** aide à préciser le sens des profils du descripteur de trafic nommés dans le paramètre d'entrée suivant. Il s'agit soit du port A, soit du port Z de la demande de connexion initiale. Le paramètre d'entrée **newNetworkCharacteristicsProfiles** indique la liste modifiée des profils associés au réseau à utiliser pour la fourniture de services où les profils à descripteur de trafic de type "b au point d'extrémité opposé" sont énumérés avant les profils à descripteur de trafic de type "point d'extrémité opposé à b". Le paramètre d'entrée **newServiceCharacteristicsProfiles** donne la liste de profils associés aux services.

La valeur de retour de type **ManagedEntityIdType** est un identificateur d'entité gérée permettant d'identifier la connexion de sous-réseau créée par suite de cette demande.

9.13.1.5 suspendService

Cette opération désactive le flux de trafic d'utilisateur au moyen de la connexion de sous-réseau de service. Pratiquement, la suspension de service est équivalente au verrouillage de l'état administratif de la connexion de sous-réseau.

La signature de l'opération **suspendService** est indiquée ci-dessous:

```
void suspendService (
            in ServiceInstanceIdType serviceInstanceId,
            in GeneralizedTimeType    startTime,
            in GeneralizedTimeType    stopTime)
            raises (UnknownServiceInstance, AccessDenied, InvalidStartTime,
            InvalidStopTime);
```

Le paramètre d'entrée **serviceInstanceId** désigne la connexion de service à suspendre. Le paramètre d'entrée **startTime** donne l'instant auquel le service est à suspendre. Le paramètre d'entrée **stopTime** donne l'instant auquel le service est à reprendre.

La valeur de retour est de type **void**.

9.13.1.6 resumeService

Cette opération active le flux de trafic d'utilisateur au moyen de la connexion de sous-réseau de service. Cette opération peut être invoquée soit après que le service a été suspendu (voir opération ci-dessus) ou lorsque la connexion de service initiale est configurée dans un état inactif. Le service est à reprendre immédiatement.

La signature de l'opération **resumeService** est indiquée ci-dessous:

```
void resumeService (
                in ServiceInstanceIdType serviceInstanceId)
                raises (UnknownServiceInstance, AccessDenied);
```

Le paramètre d'entrée **serviceInstanceId** désigne la connexion de service à reprendre.

La valeur de retour est de type **void**.

9.13.1.7 Exceptions

L'exception **AccessDenied** est déclenchée quand l'accès à l'élément de réseau est refusé pour une raison de sécurité.

L'exception **CommFailure** est déclenchée quand la communication entre le système de gestion par fournisseur et les éléments de réseau a échoué.

L'exception **ConnectionAlreadyExists** est déclenchée quand il existe déjà une connexion de sous-réseau avec les mêmes extrémités.

L'exception **ConnectionCountExceeded** est déclenchée quand le décompte de connexions dépasse le décompte admissible de connexions.

L'exception **EquipmentFailure** est déclenchée quand la connexion requise ne peut pas être appliquée à une ressource réseau installée à cause de l'état de panne de l'élément de réseau.

L'exception **InsufficientBW** est déclenchée quand la largeur de bande ne suffit pas au service requis.

L'exception **InsufficientPONBW** est déclenchée quand la largeur de bande disponible du réseau PON ne suffit pas pour prendre en charge la préfourniture requise de services.

L'exception **InvalidStartTime** est déclenchée quand l'instant de début spécifié est incompatible avec l'heure actuelle ou l'instant d'arrêt.

L'exception **InvalidStopTime** est déclenchée quand l'instant d'arrêt spécifié est incompatible avec l'heure actuelle ou l'instant de début.

L'exception **ParameterViolation** est déclenchée quand les paramètres d'extrémité ne correspondent pas aux caractéristiques protocolaires de l'accès, ou quand la ou les valeurs sont hors étendue ou des copies non valides.

L'exception **ProfileSuspended** est déclenchée quand le ou les profils nommés dans l'invocation ont été suspendus pour utilisation par le système OMS ou par l'opérateur dans le cadre du système de gestion par fournisseur.

L'exception **UnknownConnection** est déclenchée quand la connexion à supprimer n'est pas trouvée.

L'exception **UnknownNE** est déclenchée quand le nom de la terminaison OLT n'est pas connu du système de gestion par fournisseur.

L'exception **UnknownPort** est déclenchée quand l'identificateur de l'accès d'entrée n'est pas connu du système de gestion par fournisseur.

L'exception **UnknownProfiles** est déclenchée si le nom de profil fourni est inconnu du système de gestion par fournisseur et ne peut pas être extrait du répertoire d'objets de profil.

L'exception **UnknownReservationId** est déclenchée quand l'identificateur de réservation n'est pas connu du système de gestion par fournisseur.

L'exception **UnknownServiceInstance** est déclenchée quand l'identificateur de service n'est pas connu du système de gestion par fournisseur.

9.14 Module Synchroniser

Ce module gère le processus de synchronisation entre le système de gestion par fournisseur et un élément de réseau spécifique. Le processus requis de synchronisation peut être rétréci à un ensemble spécifique de listes d'événements actuels. Les détails sur la façon dont les incompatibilités sont détectées et sur la question de savoir quelles données sont extraites relèvent de l'implémentation du fournisseur. Cependant, ce processus se traduit par la suppression de toutes les incompatibilités entre les informations de gestion trouvées dans l'élément de réseau et le modèle d'information conservé dans le système de gestion par fournisseur.

Les opérations de ce module ne peuvent être invoquées que par un utilisateur privilégié. Le processus de synchronisation est de type "au mieux" dans le sens que toutes les incompatibilités détectées et corrigées avant une période de temporisation définie par le système[10] sont préservées. Avant son achèvement, le processus de synchronisation peut influencer défavorablement la qualité de fonctionnement du système de gestion par fournisseur.

Quand l'élément de réseau choisi est une terminaison OLT, alors le processus de synchronisation est effectué à l'échelle du système.

9.14.1 Interface NESynchroniser

9.14.1.1 synchNE

Cette opération lance un processus de synchronisation entre le système de gestion par fournisseur et un élément de réseau spécifique. Cette opération n'est bloquante que pendant le temps qu'il faut au système de gestion par fournisseur pour valider le fait que le processus peut être lancé et qu'il exécute l'opération de synchronisation en tâche de fond.

La signature de l'opération **synchNE** est indiquée ci-dessous:

---

[10] Cette période de temporisation relève d'un accord entre l'opérateur et le fournisseur.

```
void synchNE(in ManagedEntityIdType nEManagedEntityId)
            raises (AccessDenied, CommFailure, UnknownNE, EquipmentFailure,
            BackupInProgress, SynchInProgress);
```

L'entrée **nEManagedEntityId** désigne l'élément de réseau à synchroniser avec le système de gestion par fournisseur.

La valeur de retour est de type **void**.

9.14.1.2 abortSynchNE

Cette opération abandonne un processus de synchronisation en cours entre le système de gestion par fournisseur et un élément de réseau spécifique. Toutes les éventuelles incompatibilités entre le système de gestion par fournisseur et l'élément de réseau qui ont été résolues avant l'abandon sont conservées.

Cette opération ne peut être invoquée que par un utilisateur privilégié.

La signature de l'opération **abortSynchNE** est indiquée ci-dessous:

```
void   abortSynchNE(
             in ManagedEntityIdType nEManagedEntityId)
             raises (AccessDenied, CommFailure, UnknownNE, EquipmentFailure,
             NoSynchInProgress);
```

L'entrée **nEManagedEntityId** désigne l'élément de réseau dont la synchronisation avec le système de gestion par fournisseur est à abandonner.

La valeur de retour est de type **void**.

9.14.1.3 scheduleSynchNE

Cette opération planifie un processus de synchronisation entre le système de gestion par fournisseur et un élément de réseau spécifique. Le processus de synchronisation planifié est déclenché par le programmateur qui est prédéfini par l'opérateur. Au plus un seul programmateur peut être associé à cette activité pour un élément de réseau spécifique.

La signature de l'opération **scheduleSynchNE** est indiquée ci-dessous:

```
void scheduleSynchNE(
           in ManagedEntityIdType nEManagedEntityId,
           in UserLabelType schedulerName)
           raises (AccessDenied, UnknownNE, UnknownScheduler,
           InvalidScheduler);
```

L'entrée **nEManagedEntityId** désigne l'élément de réseau à synchroniser avec le système de gestion par fournisseur. L'entrée **schedulerName** désigne le programmateur à utiliser afin d'invoquer l'opération de synchronisation planifiée du système de gestion par fournisseur.

La valeur de retour est de type **void**.

9.14.1.4 modifyNESynchSchedule

Cette opération modifie le programme de planification pour la synchronisation d'un élément de réseau. Cette opération n'interrompra pas un processus de synchronisation en cours. En cas de succès, le nouveau programme de planification est appliqué avec l'itération suivante.

La signature de l'opération **modifyNESynch** est indiquée ci-dessous:

```
void modifyNESynchSchedule(
            in ManagedEntityIdType nEManagedEntityId,
            in UserLabelType newSchedulerName)
            raises (AccessDenied, UnknownNE, UnknownScheduler,
            InvalidScheduler);
```

L'entrée **nEManagedEntityId** désigne l'élément de réseau dans lequel la synchronisation planifiée d'élément de réseau est activée. L'entrée **newSchedulerName** désigne le programme de planification à appliquer.

La valeur de retour est de type **void**.

9.14.1.5 cancelScheduledSynchNE

Cette opération annule tous les processus de synchronisation planifiés ultérieurement pour l'élément de réseau considéré. Cette opération n'interrompra pas un processus de synchronisation en cours.

La signature de l'opération **cancelScheduledSynchNE** est indiquée ci-dessous:

```
void cancelScheduledSynchNE(
            in ManagedEntityIdType nEManagedEntityId )
            raises (AccessDenied, UnknownNE);
```

L'entrée **nEManagedEntityId** désigne l'élément de réseau dans lequel la synchronisation planifiée d'élément de réseau est activée.

La valeur de retour est de type **void**.

9.14.1.6 synchCurrentEventListings

Cette opération lance la synchronisation entre le système de gestion par fournisseur et des éléments de réseau spécifiés pour des éléments contenus dans des listes particulières d'événements actuels. Le système de gestion par fournisseur extrait normalement les valeurs actuelles d'état, de statut ou d'attributs de gestion et suit ces valeurs au moyen de listes récapitulatives d'événements actuels pour le système. Si un éventuel processus d'extraction automatique du système montre que l'énumération n'est pas à jour par rapport aux conditions actuelles du système, alors l'énumération est modifiée (par suppression d'une entrée ou insertion d'une nouvelle entrée) afin de corriger l'énumération. Cette opération est une version manuelle commandée par demande du système OMS. Pendant que cette opération est en cours, l'exception SynchInProgress ne peut pas être invoquée pour une autre opération.

La signature de l'opération **synchCurrentEventListings** est indiquée ci-dessous:

```
void synchCurrentEventListings(
                  in ManagedEntityIdType nEManagedEntityId,
                  in CurrentListingSeqType currentListingTypeList)
                  raises (AccessDenied, CommFailure, DCNTimeout, UnknownNE,
                  EquipmentFailure, Timeout);
```

L'entrée **nEManagedEntityId** désigne l'élément de réseau pour synchronisation. L'entrée **currentListingTypeList** désigne la liste des événements actuels qui doivent être synchronisés entre le système de gestion par fournisseur et des éléments de réseau spécifiés. Le système de gestion par fournisseur extrait de l'élément de réseau les valeurs spécifiées des listes d'événements actuels.

La valeur de retour est de type **void**.

9.14.1.7 scheduledSynchNEListGet

Cette opération sert à extraire les noms de tous les éléments de réseau comportant des programmes de synchronisation.

La signature de l'opération **scheduledSynchNEListGet** est indiquée ci-dessous:

```
ScheduledSynchNESeqType scheduledSynchNEListGet ()
                  raises (AccessDenied);
```

Il n'y a aucun paramètre d'entrée.

La valeur de retour est de type **ScheduledSynchNESeqType** et indique l'énumération d'éléments de réseau recherchée.

9.14.1.8 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **BackupInProgress** est déclenchée quand la demande de synchronisation est émise pendant que la sauvegarde est en cours.

L'exception **CommFailure** est déclenchée en cas de panne du RCD entre système de gestion par fournisseur et terminaison OLT ou en cas de panne de communication entre terminaison OLT et terminaisons ONT ou ONU sous-jacentes.

L'exception **DCNTimeout** est déclenchée quand la liaison de communications par RCD entre au moins un des éléments de réseau et le système de gestion par fournisseur est encombrée au point que les informations d'état actuel ou de statut ne peuvent pas être transférées dans un intervalle temporel défini par le système.

L'exception **EquipmentFailure** est déclenchée quand l'équipement où les données sont sauvegardées se trouve en état de panne.

L'exception **InvalidScheduler** est déclenchée quand le programmateur indiqué ne convient pas pour utilisation dans cette opération ou est périmé.

L'exception **SynchInProgress** est déclenchée quand une nouvelle opération synchNe est requise pendant qu'une opération synchNe ou scheduledSynchNe est en cours.

L'exception **SynchNotScheduled** est déclenchée s'il n'existe pas de synchronisation planifiée pour l'élément de réseau à l'instant spécifié.

L'exception **UnknownNE** est déclenchée quand l'élément de réseau mentionné dans la demande est inconnu du système de gestion par fournisseur.

L'exception **UnknownScheduler** est déclenchée quand le nom indiqué du programmateur n'est pas trouvé.

L'exception **Timeout** est déclenchée quand le processus spécifié a dépassé une période de temporisation par défaut définie par le système.

L'exception **NoSynchInProgress** est déclenchée s'il n'y a pas de processus de synchronisation en cours.

9.15 Module Test

Ce service sert à fournir des interfaces permettant à l'opérateur ou au système OMS d'exécuter des procédures d'essai commandées ou planifiées. Des essais tels que le rebouclage de flux cellulaires OAM en mode ATM, l'établissement d'un rebouclage d'interface sur cartes de ligne d'abonné ou sur cartes d'interface OLT réseau et les contrôles de continuité en mode ATM sont spécifiés au moyen de ce service. Les essais peuvent être soit planifiés ou invoqués manuellement à la suite d'une panne identifiée ou d'une réclamation concernant un service d'abonné. L'interface TestActionPerformer indique les opérations permettant d'exécuter des essais applicables aux ressources de réseau.

9.15.1 Interface TestActionPerformer

9.15.1.1 aTMLoopback

Cette opération sert à invoquer un essai de rebouclage de flux OAM en mode ATM. L'essai de rebouclage en mode ATM est effectué dans un seul sens.

La signature de l'opération **aTMLoopback** est indiquée ci-dessous:

```
AggregateATMLoopbackResultSeqType aTMLoopback (
                        in UserIdType testRequestorId,
                        in ManagedEntityIdType ctp,
                        in ATMLoopbackInfoType aTMLoopbackInfo,
                        in TestIterationNumType testIterationNum,
                        in ServiceInstanceIdType serviceInstanceId)
                        raises (AccessDenied, CommFailure, UnknownManagedEntity,
                        NotAvailableForTest, InvalidLocationId,
                        InvalidDirection);
```

Le paramètre d'entrée **testRequestorId** sert à identifier l'initiateur de l'essai de rebouclage en mode ATM. Le paramètre d'entrée **ctp** sert à identifier de façon unique la terminaison CTP pour le point d'injection des cellules de rebouclage. Le paramètre d'entrée **aTMLoopbackInfo** sert à fournir des informations spécifiques sur l'essai en mode ATM; l'identificateur LoopbackLocationId fait partie des informations aTMLoopbackInfo. Le paramètre d'entrée **testIterationNum** désigne le nombre d'itérations pour l'essai de rebouclage en mode ATM. Le paramètre d'entrée **serviceInstanceId** désigne l'instance de service qui peut être associée à la demande d'essai de rebouclage.

Lorsque l'on spécifie les informations aTMLoopbackInfo, les seuls valeurs applicables concernant le sens de transmission ne sont que 'egress' ou 'ingress'. Le sens est spécifié dans le cadre des informations **aTMLoopbackInfo**.

L'identificateur LoopbackLocationId désigne le ou les points d'une connexion virtuelle où le rebouclage doit se produire. La valeur par défaut de série d'unités est utilisée par l'émetteur afin d'indiquer l'extrémité. Quand l'indicateur segmentCellInd est mis à la valeur 'false', l'identificateur LoopbackLocationId doit toujours être réglé à sa valeur par défaut. Si aucun identificateur LoopbackLocationId n'est fourni, l'on part du principe qu'il existe une extrémité de segment désignée pour le flux associé au point ctp d'injection.

Pour les extrémités ctp, une série de zéros indique une demande de rebouclage orientée vers tous les points de connexion ayant un identificateur LoopbackLocationId. Une série d'unités indique une demande de rebouclage orientée vers l'extrémité (extrémité de segment ou de chaîne de connexion). 'x6A'H indique l'absence d'extrémité CP désignée pour le rebouclage, et donc qu'aucun rebouclage ne devrait être effectué. Toutes les autres valeurs de l'identificateur LoopbackLocationId indiquent une demande de rebouclage orientée vers un emplacement à un identificateur LoopbackLocationId spécifique.

La valeur de retour est de type **AggregateATMLoopbackResultSeqType** qui donne des informations sur l'essai, spécifiquement l'identificateur loopbackingLLID, le temps de réponse en microsecondes, et succès/échec pour chaque itération.

9.15.1.2 initializeContinuityCheck

Cette opération sert à préparer un contrôle de continuité ATM. La création d'un environnement d'essai de continuité ne lance pas forcément l'essai, mais planifie plutôt le lancement d'un essai. Après création réussie d'un contrôle de continuité, le système renvoie un unique identificateur qui sert à identifier l'essai.

Cette opération ne se rapporte qu'à l'établissement de l'essai. Une fois que l'essai est exécuté, une alarme est déclenchée en cas d'échec.

La signature de l'opération **initializeContinuityCheck** est indiquée ci-dessous:

```
CCSetUpIdType initializeContinuityCheck(
                  in UserIdType testRequestorId,
                  in ManagedEntityIdType sourceCtp,
                  in ATMContinuityCheckInfoType aTMContinuityCheckInfo,
                  GeneralizedTimeType   stopTime,
                  in ServiceInstanceIdType serviceInstanceId)
```

```
                raises (AccessDenied, CommFailure, UnknownManagedEntity,
                NotAvailableForTest, InvalidStartTime, InvalidStopTime,
                InvalidDirection);
```

Le paramètre d'entrée **testRequestorId** sert à identifier l'initiateur de l'essai. Les paramètres d'entrée **sourceCtp** et **sinkCtp** servent à identifier le point A pour l'essai de continuité. Le paramètre d'entrée **aTMContinuityCheckInfo** sert à fournir des informations spécifiques sur l'essai de continuité – le sens doit toujours être mis à 'BothDirections' et l'indicateur segmentCellInd est mis à la valeur 'true' pour l'essai de continuité d'un segment et à la valeur 'false' pour l'essai de bout en bout. Le paramètre d'entrée **stopTime** donne des informations sur la durée pendant laquelle l'essai devrait être exécuté. Le paramètre d'entrée **serviceInstanceId** désigne l'instance de service qui peut être associée à la demande d'essai de rebouclage.

La valeur de retour est de type **CCSetUpIdType** et désigne de façon unique l'essai qui a été établi. L'identificateur CCSetUpId pour l'essai établi existe jusqu'à ce que l'instant d'arrêt ait été atteint ou jusqu'à ce qu'il soit explicitement annulé au moyen de l'opération terminateContinuityCheck.

9.15.1.3 terminateContinuityCheck

Cette opération sert à supprimer l'environnement d'essai de continuité ATM. Si l'essai de continuité a déjà été lancé (c'est-à-dire si son instant de début a été atteint), l'essai de continuité arrêtera de s'exécuter et sera ensuite supprimé.

La signature de l'opération **terminateContinuityCheck** est indiquée ci-dessous:

```
void terminateContinuityCheck(
                in CCSetUpIdType   cCSetUpId)
                raises (AccessDenied, CommFailure, UnknownTest);
```

Le paramètre d'entrée **CCSetUpId** sert à identifier l'essai à terminer.

La valeur de retour est de type **void**.

9.15.1.4 scheduleResourceSelfTest

Cette opération sert à programmer dans le temps un essai d'autovérification d'une ressource. Cette opération est utilisée par le système OMS afin d'établir des essais d'autovérification sur ressources à exécuter régulièrement. L'établissement d'un objet de programmateur est un prérequis pour lancer cette opération.

La signature de l'opération **scheduleResourceSelfTest** est indiquée ci-dessous:

```
TestTrackingObjectIdType scheduleResourceSelfTest(
              in UserIdType testRequestorId,
              in ManagedEntityIdType targetNE,
              in unsigned long timeOutPeriod, //In seconds.
              in ResourceSelfTestInfoSeqType specificTestInfo,
              in UserLabelType schedulerName)
              raises (AccessDenied, UnknownNE, UnknownScheduler,
              InvalidScheduler, InvalidTimeoutPeriod, InvalidTestOperations);
```

Le paramètre d'entrée **testRequestorId** sert à identifier l'initiateur de l'essai. Le paramètre d'entrée **targetNE** désigne l'élément de réseau qui doit exécuter l'essai d'autovérification. Le paramètre d'entrée **timeOutPeriod** désigne la durée maximale pendant laquelle le système permettra l'exécution de l'essai sur la ressource. Le paramètre d'entrée **specificTestInfo** sert à fournir des informations propres au fournisseur sur l'essai d'autovérification concernant chaque type d'essai de diagnostic à exécuter; ces informations sont fournies à l'opérateur au moyen de la documentation du système. Le paramètre d'entrée **schedulerName** sert à faire référence au programmateur applicable à cet essai.

La valeur de retour de type **testTrackingObjectIdType** désigne de façon unique les essais planifiés. L'objet de suivi d'essai existe jusqu'à ce qu'il soit explicitement annulé au moyen de l'opération terminateScheduledResourceSelfTest ou lorsque l'instant d'arrêt du programmateur a été atteint.

Les résultats de l'essai d'autovérification sont journalisés. Le type de données **ResourceSelfTestResultSeqType** définit une partie des informations qui sont journalisées.

9.15.1.5 modifyResourceSelfTestSchedule

Cette opération sert à modifier le programme de planification pour un essai d'autovérification de ressource conduit régulièrement. En cas de succès, le lancement de l'essai d'autovérification d'une ressource est changé avec l'itération suivante.

La signature de l'opération **modifyResourceSelfTestSchedule** est indiquée ci-dessous:

```
void   modifyResourceSelfTestSchedule (
             in TestTrackingObjectIdType testTrackingObjectId,
             in UserLabelType    newSchedulerName)
             raises (UnknownTest, UnknownScheduler, InvalidScheduler,
             AccessDenied);
```

Le paramètre d'entrée **testTrackingObjectId** sert à identifier l'essai planifié. Le paramètre d'entrée **testTrackingObjectId** sert à identifier le nouveau programme de planification.

La valeur de retour est de type **void**.

9.15.1.6 cancelScheduledResourceSelfTest

Cette opération sert à annuler un essai régulièrement planifié d'autovérification d'une ressource. En cas de succès, cette opération annule l'essai avant son lancement au prochain instant de déclenchement.

La signature de l'opération **cancelScheduledResourceSelfTest** est indiquée ci-dessous:

```
void  cancelScheduledResourceSelfTest (
            in TestTrackingObjectIdType testTrackingObjectId)
            raises (UnknownTest, UncontrolledTestInProgress, AccessDenied);
```

Le paramètre d'entrée **testTrackingObjectId** sert à identifier l'essai planifié à annuler.

La valeur de retour est de type **void**.

9.15.1.7 conductResourceSelfTest

Cette opération sert à lancer un essai d'autovérification d'une ressource après l'identification d'une panne de système ou une réclamation concernant un service d'abonné. Les résultats d'essai sont consignés dans le système de gestion par fournisseur.

La signature de l'opération **conductResourceSelfTest** est indiquée ci-dessous:

```
TestTrackingObjectIdType conductResourceSelfTest (
              in UserIdType testRequestorId,
              in ManagedEntityIdType targetNE,
              in unsigned long timeOutPeriod, //In seconds.
              in ResourceSelfTestInfoSeqType specificTestInfo)
              raises (AccessDenied, CommFailure, UnknownNE,
              InvalidTimeoutPeriod, InvalidTestOperations);
```

Le paramètre d'entrée **testRequestorId** sert à identifier l'initiateur de l'essai. Le paramètre d'entrée **targetNE** désigne l'élément de réseau qui doit exécuter l'essai d'autovérification. Le paramètre d'entrée **timeOutPeriod** désigne la période où le système essaiera de lancer l'essai. Le paramètre d'entrée **specificTestInfo** sert à fournir des informations spécifiques sur l'essai d'autovérification et indique des détails sur chaque type d'essai de diagnostic à exécuter.

La valeur de retour est de type **testTrackingObjectId** et indique un mécanisme afin de permettre à l'opérateur de mettre fin à un essai commandé d'autovérification de ressource. Si l'essai d'autovérification d'une ressource n'est pas commandé, le système de gestion par fournisseur renvoie la valeur 0.

9.15.1.8 terminateResourceSelfTest

Cette opération termine un essai d'autovérification d'une ressource en cours.

La signature de l'opération **terminateResourceSelfTest** est indiquée ci-dessous:

```
ResourceSelfTestResultSeqType terminateResourceSelfTest (
            in TestTrackingObjectIdType testTrackingObjectId)
            raises (UnknownTest, UncontrolledTestInProgress, AccessDenied);
```

Le paramètre d'entrée **testTrackingObjectId** sert à identifier l'essai à annuler.

La valeur de retour de type **ResourceSelfTestResultSeqType** indique les éventuels résultats d'essai intérimaires qui sont disponibles**.**

9.15.1.9 initiateLoopback

Cette opération sert à lancer un rebouclage pour un service. Par exemple, l'on exécutera un rebouclage de ligne numérique DS1 à l'extrémité locale [afin de pouvoir prendre en charge l'extrémité distante] ou un raccordement à une ressource SONET.

La signature de l'opération **initiateLoopback** est indiquée ci-dessous:

```
LoopbackTrackingObjectIdType    initiateLoopback (
                                in UserIdType testRequestorId,
                                in ManagedEntityIdType loopingCtp,
                                in long duration, //In minutes.
                                in DirectionalityType directionality,
                                in LoopbackTestType loopbackTest
                                in ServiceInstanceIdType serviceInstanceId)
                                raises (AccessDenied, CommFailure,
                                UnknownManagedEntity, NotAvailableForTest);
```

Le paramètre d'entrée **testRequestorId** sert à identifier l'initiateur de l'essai. Le paramètre d'entrée **loopingCtp** sert à identifier la terminaison CTP sur laquelle le rebouclage doit être effectué. Le paramètre d'entrée **duration** définit la durée en secondes pendant laquelle le rebouclage devrait être actif. Le paramètre d'entrée **directionality** indique si le rebouclage devrait être effectué pour le trafic entrant, le trafic sortant ou pour le trafic dans les deux sens. Le paramètre d'entrée **loopbackTest** sert à identifier le type spécifique d'essai de rebouclage. Le paramètre d'entrée **serviceInstanceId** sert à spécifier le service associé à cette opération de rebouclage.

La valeur de retour de type **LoopbackTrackingObjectIdType** désigne de façon unique l'essai de rebouclage qui est lancé. Une fois qu'un rebouclage a terminé son cycle d'exécution (c'est-à-dire à la fin de la durée de l'essai), l'objet n'est plus disponible.

9.15.1.10 terminateLoopback

Cette opération sert à annuler un rebouclage en cours.

La signature de l'opération **terminateLoopback** est indiquée ci-dessous:

```
void terminateLoopback  (
                        in LoopbackTrackingObjectId loopbackTrackingObjectId)
                        raises (UnknownTest, AccessDenied);
```

Le paramètre d'entrée **loopbackTrackingObjectId** sert à identifier le rebouclage à annuler.

La valeur de retour est de type **void**.

9.15.1.11 getLoopbackInfo

Cette opération sert à extraire les informations de rebouclage d'une extrémité de connexion particulière.

La signature de l'opération **getLoopbackInfo** est indiquée ci-dessous:

```
LoopbackInfoType getLoopbackInfo (
                        in ManagedEntityIdType cTP)
                        raises (UnknownManagedEntity, AccessDenied);
```

Le paramètre d'entrée **cTP** sert à identifier l'emplacement possible du rebouclage.

La valeur de retour est de type **LoopbackInfoType** et elle spécifie le type et le sens du rebouclage.

9.15.1.12 getLoopbackInfoByNE

Cette opération sert à extraire l'emplacement et les détails de chaque point de connexion qui est en mode de rebouclage dans un élément de réseau.

La signature de l'opération **getLoopbackInfoByNE** est indiquée ci-dessous:

```
LoopbackInfoSeqType getLoopbackInfoByNE    (
                        in ManagedEntityIdType nEId)
                        raises (UnknownManagedEntity, AccessDenied);
```

Le paramètre d'entrée **nEId** sert à identifier la ressource réseau.

La valeur de retour de type **LoopbackInfoSeqType** indique une énumération d'emplacements, de types et de sens de rebouclage.

9.15.1.13 getTestStatus

Cette opération sert à extraire l'état d'un rebouclage en cours.

La signature de l'opération **getTestStatus** est indiquée ci-dessous:

```
StatusValueType getTestStatus (
                    in LoopbackTrackingObjectIdType   id)
                    raises (AccessDenied, UnknownTest);
```

Le paramètre d'entrée **id** spécifie l'établissement de l'essai considéré.

La valeur de retour est de type **StatusValueType** et indique l'état de l'action de rebouclage.

9.15.1.14 scheduledTestNEListGet

Cette opération sert à extraire la liste de tous les essais planifiés d'un élément de réseau.

La signature de l'opération **scheduledTestNEListGet** est indiquée ci-dessous:

```
ScheduledTestNESeqType scheduledTestNEListGet ()
                    raises (AccessDenied);
```

Il n'y a aucun paramètre d'entrée.

La valeur de retour est de type **scheduledTestNESeqType** et indique l'énumération recherchée.

9.15.1.15 testHistoryByManagedEntity

Cette opération sert à extraire l'historique d'essai associé à une entité gérée. Cet historique devrait être conservé aussi longtemps que requis par l'opérateur.

La signature de l'opération **testHistoryByManagedEntity** est indiquée ci-dessous:

```
TestHistorySeqType testHistoryByManagedEntity (
                   in ManagedEntityIdType managedEntityId)
                   raises (AccessDenied, UnknownManagedEntity);
```

Le paramètre d'entrée **managedEntityId** spécifie l'entité gérée de référence.

La valeur de retour est de type **testHistorySeqType** et indique l'énumération recherchée.

9.15.1.16 testHistoryByServiceInstance

Cette opération sert à extraire l'historique d'essai associé à une instance de service. Cet historique devrait être conservé aussi longtemps que requis par l'opérateur.

La signature de l'opération **testHistoryByServiceInstance** est indiquée ci-dessous:

```
TestHistorySeqType testHistoryByServiceInstance (
                   in ServiceInstanceIdType serviceInstanceId)
                   raises (AccessDenied, UnknownServiceInstance);
```

Le paramètre d'entrée **serviceInstanceId** spécifie le référence instance de service.

La valeur de retour est de type **testHistorySeqType** et indique l'énumération recherchée.

9.15.1.17 Exceptions

L'exception **AccessDenied** est déclenchée quand le système n'est pas autorisé à accéder à l'objet d'interface.

L'exception **CommFailure** est déclenchée en cas de panne du RCD entre système de gestion par fournisseur et terminaison OLT ou de panne de communication entre terminaison OLT et terminaison ONT de départ.

L'exception **InvalidDirection** est déclenchée quand le sens spécifié de l'essai est non valide pour l'essai considéré.

L'exception **InvalidLocationId** est déclenchée quand l'identificateur LLID spécifié n'est pas valide.

L'exception **InvalidScheduler** est déclenchée quand le programmateur indiqué ne convient pas pour utilisation dans cette opération ou est périmé.

L'exception **InvalidStartTime** est déclenchée quand l'instant de début spécifié est incompatible avec la matrice actuelle des déclencheurs temporels ou l'instant d'arrêt.

L'exception **InvalidStopTime** est déclenchée quand l'instant d'arrêt spécifié est incompatible avec la matrice actuelle des déclencheurs temporels ou l'instant de début.

L'exception **InvalidTestOperations** est déclenchée quand l'opération requise d'autovérification n'est pas valide.

L'exception **InvalidTimeOutPeriod** est déclenchée quand la période de temporisation désignée viole la définition de valeurs valides.

L'exception **NotAvailableForTest** est déclenchée quand la terminaison CTP de départ n'est pas capable d'établir un établissement d'essai de vérification de continuité avec la terminaison CTP de destination.

L'exception **UncontrolledTestInProgress** est déclenchée quand l'essai d'autovérification ne peut pas être annulé en raison d'un essai non commandé.

L'exception **UnknownNE** est déclenchée quand l'élément de réseau mentionné dans la demande est inconnu du système de gestion par fournisseur.

L'exception **UnknownManagedEntity** est déclenchée quand l'entité gérée spécifiée est inconnue du système de gestion par fournisseur.

L'exception **UnknownServiceInstance** est déclenchée quand l'instance de service spécifiée est inconnue du système de gestion par fournisseur.

L'exception **UnknownScheduler** est déclenchée quand le nom indiqué du programmateur n'est pas trouvé.

L'exception **UnknownTest** est déclenchée quand l'essai spécifié par l'identificateur de suivi n'est pas connu dans le système de gestion par fournisseur.

9.16 Module FileTransfer

Ce module se rapporte à la gestion des articles enregistrés de transfert en temps planifié qui sont mémorisés dans toute archive à court terme du système de gestion par fournisseur. Il prend en charge le suivi et la surveillance ultérieurs du processus de transfert de fichiers en utilisant l'identificateur d'objet de suivi de transfert. Le système de gestion par fournisseur journalise les résultats favorables ou défavorables du transfert de fichiers. Toute demande de transfert de fichiers est accompagnée par des justificatifs de sûreté par lesquels le système de gestion par fournisseur est autorisé à communiquer avec le serveur de destination. Le transfert de fichiers peut être planifié par avance. Les objets de suivi de transfert sont automatiquement supprimés par le système de gestion du fournisseur une fois que le transfert de fichiers associé est terminé et les résultats (échec ou succès) sont enregistrés dans le journal d'exécution.

9.16.1 Interface TransferMgr

9.16.1.1 fileTransfer

Le transfert de fichiers est lancé immédiatement par cette opération.

La signature de l'opération **fileTransfer** est indiquée ci-dessous:

```
TransferTrackingObjectIdType fileTransfer(
                        in ManagedEntityIdType recordSetId,
                        in DCNAddressType destinationServerAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in FilenameType destinationFile,
                        in boolean overwriteExistingFile)
                        raises (AccessDenied, CommFailure, UnknownRecordSet,
                        UnknownDestinationServer );
```

Le paramètre d'entrée **recordSetId** désigne l'archive à court terme de laquelle les données doivent être extraites pour transfert de fichiers. Le paramètre d'entrée **destinationServerAddr** désigne l'adresse de mise en réseau de communication de données (RCD) pour le serveur qui est la destination du transfert de fichiers. Les paramètres d'entrée **userId** et **password** fournissent le mécanisme d'ouverture de session au serveur de destination (en supposant que de tels justificatifs de sûreté sont requis). Le paramètre d'entrée **destinationFile** indique un répertoire d'accueil entier pour le fichier transféré. Finalement, le paramètre **overwriteExistingFile** indique si le transfert de fichiers devrait permettre l'écrasement d'un fichier préexistant ayant le même emplacement de répertoire d'accueil.

La valeur de retour est de type **TransferTrackingObjectIdType** et donne une clé de corrélation à utiliser lors d'une tentative de suivi de l'état du transfert en temps planifié de données déjà extraites des archives à court terme.

9.16.1.2 scheduleFileTransfer

Le transfert de fichiers est planifié pour lancement futur par cette opération. Tout le contenu de l'archive à court terme désignée est extrait pour transfert de fichiers. Aucune hypothèse n'est faite concernant la purge de l'archive après un transfert de fichiers de données réussi. En revanche,

l'archive est purgée en fonction d'accords entre le fournisseur et l'opérateur concernant la politique de conservation des informations archivées.

La signature de l'opération **scheduleFileTransfer** est indiquée ci-dessous:

```
TransferTrackingObjectIdType scheduleFileTransfer (
                        in ManagedEntityIdType recordSetId,
                        in DCNAddressType destinationServerAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in FilenameType destinationFile,
                        in boolean overwriteExistingFile,
                        in UserLabelType schedulerName)
                        raises (AccessDenied,UnknownRecordSet,
                        UnknownDestinationServer, UnknownScheduler,
                        InvalidScheduler);
```

Le paramètre d'entrée **recordSetId** désigne l'archive à court terme de laquelle les données doivent être extraites pour transfert de fichiers. Le paramètre d'entrée **destinationServerAddr** désigne l'adresse de mise en réseau de communication de données (RCD) pour le serveur qui est la destination du transfert de fichiers. Les paramètres d'entrée **userId** et **password** fournissent le mécanisme d'ouverture de session au serveur de destination (en supposant que de tels justificatifs de sûreté sont requis). Le paramètre d'entrée **destinationFile** indique un répertoire d'accueil entier pour le fichier transféré. Le paramètre **overwriteExistingFile** indique si le transfert de fichiers devrait permettre l'écrasement d'un fichier préexistant ayant le même emplacement de répertoire d'accueil. Finalement, le paramètre d'entrée **schedulerName** est l'information de planification associée au transfert de fichiers, sur la base de laquelle ce transfert intervient.

La valeur de retour est de type **TransferTrackingObjectIdType** et donne une clé de corrélation à utiliser lors d'une tentative de suivi ultérieur de l'état du transfert en temps planifié de données de l'archive à court terme.

9.16.1.3 modifyFileTransferSchedule

Cette opération modifie le programme de transfert de fichiers. En cas de succès, le nouveau programme est appliqué avec l'itération suivante.

La signature de l'opération **modifyFileTransferSchedule** est indiquée ci-dessous:

```
void modifyFileTransferSchedule (
                        in TransferTrackingObjectIdType
                        transferTrackingObjectId,
                        in UserLabelType newSchedulerName)
                        raises (AccessDenied, UnknownTransferProcess,
                        UnknownScheduler, InvalidScheduler);
```

Le paramètre d'entrée **transferTrackingObjectId** désigne l'activité planifiée de transfert de fichiers. Le paramètre d'entrée **newSchedulerName** est la nouvelle information sur le programme de planification à associer au transfert de fichiers.

La valeur de retour est de type **void**.

9.16.1.4 cancelScheduledFileTransfer

Cette opération annule le transfert de fichiers. En cas de succès, l'activité est annulée avec l'itération suivante.

La signature de l'opération **cancelScheduledFileTransfer** est indiquée ci-dessous:

```
void cancelScheduledFileTransfer (
                  in TransferTrackingObjectIdType transferTrackingObjectId)
                  raises (AccessDenied, UnknownTransferProcess);
```

Le paramètre d'entrée **transferTrackingObjectId** désigne l'activité planifiée de transfert de fichiers.

La valeur de retour est de type **void**.

9.16.1.5 getStatus

Cette opération permet au client de vérifier l'état d'un transfert avant son achèvement au moyen d'une clé.

La signature de l'opération **getStatus** est indiquée ci-dessous:

```
StatusValueType getStatus (
                  in TransferTrackingObjectIdType transferTrackingObjectId)
                  raises (UnknownTransferProcess, AccessDenied);
```

Le paramètre d'entrée **transferTrackingObjectId** désigne la clé d'un processus de transfert de fichiers particulier. L'opérateur spécifie ces informations et trouve les informations d'état d'avancement de ce processus de transfert de fichiers.

La valeur de retour de type **StatusValueType** donne l'état d'avancement du processus de transfert de fichiers.

9.16.1.6 fileTransferHistoryListGet

Cette opération sert à extraire une liste de tous les transferts de fichiers effectués pour le système de gestion par fournisseur. Cette liste est conservée dans le système de gestion par fournisseur sous la forme d'un journal à débordement.

La signature de l'opération **fileTransferHistoryListGet** est indiquée ci-dessous:

```
FileTransferHistorySeqType fileTransferHistoryListGet ()
                  raises (AccessDenied);
```

Il n'y a aucun paramètre d'entrée.

La valeur de retour est de type **FileTransferHistorySeqType** et indique l'énumération recherchée.

9.16.1.7 scheduledFileTransferListGet

Cette opération sert à extraire les noms de tous les transferts de fichiers déjà planifiés qui sont définis pour le système de gestion par fournisseur.

La signature de l'opération **scheduledFileTransferListGet** est indiquée ci-dessous:

```
ScheduledFileTransferSeqType scheduledFileTransferListGet ()
                  raises (AccessDenied);
```

Il n'y a aucun paramètre d'entrée.

La valeur de retour est de type **scheduledFileTransferSeqType** et indique l'énumération recherchée.

9.16.1.8 Exceptions

L'exception **AccessDenied** est déclenchée quand le client n'est pas autorisé à accéder à cet objet d'interface.

L'exception **CommFailure** est déclenchée quand il existe une panne de communication entre le serveur de destination et le système de gestion par fournisseur.

L'exception **UnknownDestinationServer** est déclenchée quand le serveur de destination spécifié ne peut pas être consulté par l'agent de transfert.

L'exception **UnknownRecordSet** est déclenchée quand le fichier cible ne peut pas être trouvé.

L'exception **UnknownScheduler** est déclenchée quand le programmateur spécifié ne peut pas être consulté par l'agent de transfert.

L'exception **UnknownTransferProcess** est déclenchée quand l'identificateur spécifié d'objet de suivi de transfert **TransferTrackingObjectId** ne peut pas être identifié.

10 Déclaration de conformité

Une implémentation revendiquant la conformité à l'une des interfaces définies dans la présente Recommandation doit implémenter l'ensemble du comportement associé à toutes les opérations de cette interface ainsi que le comportement des définitions citées en référence dans le module q834_4:: Q834Common.

Annexe A

Dictionnaire de données

Le Tableau A.1 donne une énumération complète de tous les éléments de données (types de données) utilisés dans la spécification d'interface contenue dans la présente Recommandation. Cette énumération comprend l'interprétation des informations de gestion, leur syntaxe et tous commentaires qualificatifs pour chaque élément. Les éléments de données sont énumérés en ordre alphabétique. Si un type de données et un second type de données construit en tant que séquence du premier sont tous deux présents dans la spécification d'interface, seul le premier élément de données est défini. La définition du second élément est évidente. Dans ces cas, le nom de la séquence est le nom de l'élément de données initial avec les caractères "Seq" insérés avant le "Type" final".

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`AAL1PMHistoryDataType`	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	`struct`	
`AAL1ProfileType`	Cet élément de données donne les valeurs pour un profil de sorte AAL1.	`struct`	
`AAL2PMHistoryDataType`	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	`struct`	
`AAL2ProfileType`	Cet élément de données donne les valeurs pour un profil de sorte AAL2.	`struct`	
`AAL2PVCProfileType`	Cet élément de données donne les valeurs pour un profil de sorte AAL2PVC.	`struct`	
`AAL5PMHistoryDataType`	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	`struct`	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
AAL5ProfileType	Cet élément de données donne les valeurs pour un profil de sorte AAL5.	struct	
AALModeType	Cet élément de données indique le mode employé par la couche AAL pour la connexion VCC sous-jacente.	enum	
ActivityLevelType	Spécifie le niveau d'autorisation d'accès apporté à un utilisateur pour une activité.	enum	monitorOnly, allowedTo Execute, noAccess
ActivityType	Spécifie le type ou la catégorie de l'activité d'utilisateur.	short	Elément défini par des constantes dans l'interface q834_4::Access Control::AccessC ontrolMgr
AdministrationDomain Type	Identificateur fourni par le système OMS ou par l'opérateur pendant l'inscription afin d'indiquer le domaine administratif auquel l'élément de réseau appartient.	UserLabelType	
AdministrativeState Type	Sert à activer (déverrouiller), à désactiver (verrouiller), ou à fermer (clore) les fonctions de l'entité gérée associée.	enum	Défini dans la Rec. UIT-T X.780
AggregateATMLoopback ResultSeqType	Spécifie le résultat composite de l'essai de rebouclage en mode ATM.	struct	
AlarmLogRecordType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
AlarmStatusSeqType	Cet élément de données donne toutes les valeurs applicables à l'état d'alarme variable.	enum	Les valeurs valides d'énumération sont: AS_UnderRepair, AS_Critical, AS_Major, AS_Minor, AS_Alarm Outstanding
AnnouncementType	Cet élément de données donne l'annonce au client qui décroche quand aucune tentative d'appel n'a eu lieu.	enum	
APONPMHistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
AppIdType	Cet élément de données spécifie les combinaisons de protocoles utilisées entre les fonctions d'interfonctionnement situées dans la fonction de passerelle vocale et la terminaison ONT ou NT.	enum	
ATMContinuityCheckInfo Type	Spécifie l'entrée pour le contrôle de continuité ATM.	struct	
ATMLoopbackInfoType	Spécifie les informations d'essai de rebouclage en mode ATM.	struct	
ATMNetworkAccess ProfileType	Profile une instance de profil de la sorte ATMNetworkAccess.	struct	
ATMOverbookingFactor ProfileType	Cet élément de données donne les valeurs pour un profil de sorte Facteur de surréservation ATM.	struct	
AudioServIndType	Cet élément de données booléen indique si le service audio est transporté, la valeur TRUE impliquant la présence de ce service.	boolean	
AutoDetectionIndType	Cet élément de données booléen indique si l'autodétection du débit de données est activée.	boolean	
AvailableSysBandwidth SeqType	Enumération des largeurs de bande du système disponibles à chaque port	Séquence de PortBandwidth Type	
AvailabilityStatusSet Type	Cet élément de données donne toutes les valeurs applicables à l'état de disponibilité variable.	Séquence de valeurs d'état de disponibilité	Défini dans la Rec. UIT-T X.780
BackedUpStatusType	Cet élément de données indique si l'entité gérée qui émet l'alarme a une unité de sauvegarde opérationnelle.	boolean	Défini dans la Rec. UIT-T X.780
BridgedLANService ProfileType	Cet élément de données donne les valeurs pour un profil de sorte Service LAN routé.	struct	
BridgePriorityType	Cet élément de données indique si l'entité gérée qui émet l'alarme a une unité de sauvegarde opérationnelle.	short	
BRISignallingType	Cet élément de données sélectionne quel format de signalisation devrait être utilisé pour le RNIS au débit de base.	enum	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
BufferedCDVTolerance Type	Cet élément de données représente la durée des données d'utilisateur qui doivent toujours être mises en mémoire tampon par l'entité d'interfonctionnement de service CES afin de compenser la variation du temps de transfert cellulaire. Ce rythme sera par incréments de 10 µs.	long long	
CableLengthType	Cet élément de données donne la longueur de câble à paire torsadée de la terminaison physicalPathTP de l'interface de type "DS1" jusqu'au point de brassage DSX1 (si applicable).	long long	
CASType	Cet élément de données sélectionne le format AAL1 qui devrait être utilisé. Il ne s'applique qu'à des interfaces structurées. Dans les interfaces non structurées, cet élément, si présent, doit toujours être réglé à la valeur par défaut du système de base.	enum	
CASIndType	Cet élément de données booléen indique si La signalisation voie par voie est activée sur la connexion, la valeur TRUE impliquant son activation.	boolean	
CBRRateType	Cet élément de données représente le débit du service de débit CBR pris en charge par la couche AAL.	enum	
CCSetUpIdType	Spécifie un unique Id pour montage d'essai CC.	long long	
CDVTPCREgressType	Tolérance sur la variation du temps de transfert cellulaire – ce paramètre est requis pour toutes les catégories de service. Il s'applique aux flux à priorité CLP = 0 pour le débit ABR et aux flux à priorité CLP = 0 + 1 sinon.	long long	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
CDVTPCRIngressType	Tolérance sur la variation du temps de transfert cellulaire – ce paramètre est requis pour toutes les catégories de service. Il s'applique aux flux à priorité CLP = 0 pour le débit ABR et aux flux à priorité CLP = 0 + 1 sinon.	long long	
CellLossIntegration PeriodType	Cet élément de données représente la durée en millisecondes pour la période d'intégration de perte de cellules. Si des cellules sont perdues pendant cet intervalle de temps, l'entité d'interfonctionnement associée VcCTPF provoquera une alarme de disette cellulaire.	long long	
CESServiceProfileType	Cet élément de données donne les valeurs pour un profil de sorte Service d'émulation CES.	struct	
ClockRecoveryType	Cet élément de données indique si le type de récupération du rythme est issu de l'interface physique.	enum	
CMDataIndType	Cet élément de données booléen indique si les données en mode circuit sont transportées sur cette connexion, la valeur TRUE impliquant sa présence.	boolean	
CMMultiplierNumType	Cet élément de données donne la valeur *N* dans les données en mode circuit à $N \times 64$ kbit/s.	short	
ConformanceDefType	Indique le type de la conformité comme défini dans ATM-Forum TM 4.0.	enum	
ControlStatusSetType	Cet élément de données donne toutes les valeurs applicables à l'état de commande variable.	Séquence de: enum	Défini dans la Rec. UIT-T X.780.
CorrelatedNotification Type	Cet élément de données énumère les numéros de référence d'autres notifications d'évènement qui ont une relation avec cette notification d'évènement.	Séquence de: Notification Identifier Type	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`CreationModeType`	Cet élément de données indique comment la chaîne d'articles a été créée.	`enum`	Choix entre types définis par l'opérateur ou types instanciés dans le cadre de l'instal-lation d'une application du système de gestion par fournisseur.
`CurrentListingType`	Spécifie les listes d'évènements actuels qui ont besoin d'être synchronisées entre le système de gestion par fournisseur et l'élément de réseau.	`short`	Valeurs spécifiées comme constantes.
`CurrentSizeType`	Cet élément de données décrit la longueur actuelle d'une chaîne d'articles.	`unsigned long long`	Dans les mêmes unités que `MaxSizeType`
`DataRateType`	Cet élément de données donne le débit binaire de la connexion Ethernet. Les valeurs valides sont 10 Mbit/s ou 100 Mbit/s.	`enum`	
`DayOfMonthType`	Spécifie le jour du mois.	`short`	0 est interprété comme signifiant: non spécifié.
`DayOfWeekType`	Spécifie le jour de la semaine dans le programme de planification.	`enum`	Dimanche, Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, non spécifié
`DCNAddressType`	Fournit l'adresse de l'élément de réseau ou du serveur de système dans le réseau de communication de données de l'opérateur. Utilisé pour le routage de messages.	`string`	Normalement: adresse IP. Exemples: items étiquetés par `softwareSource Addr`, `destination ServerAddr`, `sourceServer Addr`, `nEDCNAddress` et `newNEDCNAddress`
`DefaultSSCSParameter Profile1PtrType`	Cet élément de données désigne les valeurs par défaut pour le profil de service de convergence propre au service associé à des voies acheminant le trafic du plan de commande et du plan de gestion.	`Name`	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`DefaultSSCSParameter Profile2PtrType`	Cet élément de données désigne les valeurs par défaut pour le profil de service de convergence propre au service associé à des voies carrying media streams (par exemple, POTS ou RNIS B-channels).	`Name`	
`DiagnosticType`	Spécifie le type d'essai de diagnostic.	`short`	Propre au fournisseur
`DirectionalityType`	Spécifie le sens de l'essai.	`enum`	Valeurs: `"Egress, Ingress, Both Directions"`
`DirectionType`	Spécifie le sens de la longueur d'onde optique associée à un accès.	`enum`	Valeurs: "unidirection" ou "bidirection".
`DownloadStatusSeqType`	Cet élément de données donne l'état de toutes les activités de téléchargement de logiciels associées à une demande initiale d'acheminement de logiciels.	séquence de: structures composée de l'Id d'une cible suivi par l'état d'achemine-ment, de distribution, de validation, et d'activation.	
`DS1EncodingType`	Cet élément de données désigne le type de codage DS1 en ligne.	`enum`	
`DS1FramingType`	Cet élément de données désigne le type de verrouillage de trames employé.	`enum`	
`DS1PMHistoryDataType`	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	`struct`	
`DS1ProfileType`	Cet élément de données donne les valeurs pour un profil de sorte DS1.	`struct`	
`DS3ApplicationType`	Cet élément de données désigne le type de signal DS3.	`enum`	
`DS3PMHistoryDataType`	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	`struct`	
`DS3ProfileType`	Cet élément de données donne les valeurs pour un profil de sorte DS3.	`struct`	
`DS3EncodingType`	Cet élément de données désigne le codage DS3 en ligne.	`enum`	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
DTEDCEType	Cet élément de données indique si le câblage d'interface Ethernet est en ETTD ou en ETCD.	boolean	
DTMFIndType	Cet élément de données booléen indique si les chiffres de numérotation DTMF sont transportés sur la connexion, la valeur TRUE impliquant sa présence.	boolean	
DuplexType	Cet élément de données indique si le mode de transmission bilatérale simultanée (=TRUE) ou à l'alternat (=FALSE) est employé.	boolean	
E1PMHistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
E3PMHistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
EchoCancellationIndType	Cet élément de données indique la présence de circuits d'annulation d'écho.	boolean	
ELCPIndType	Cet élément de données booléen indique si le protocole de commande de circuit émulé est en usage.	boolean	
EncapsulationProtocolType	Cet élément de données désigne le protocole d'encapsulation utilisé pour router un réseau LAN vers le mode ATM.	short	
EncProfileIndexType	Cet élément de données indique le profil spécifique de codage prédéfini à utiliser.	short	
EncSrcTypeType	Cet élément de données indique l'origine pour le format du profil de codage. Les valeurs valides comprennent mais ne sont pas limitées à: "UIT-T" et "ATM Forum".	enum	
EndPointType	Cet élément spécifie les caractéristiques d'une extrémité de connexion	struct	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
EquipmentHolderAddressType	Cet élément de données désigne l'emplacement d'équipement pour une carte d'équipement de ligne.	structure dont les composants comprennent le numéro d'alvéole (court) et le numéro d'accès (court)	
EquipmentHolderFType	Cette structure de données désigne l'item equipmentHolderF nommé dans l'évènement d'autodécouverte.	structure énumérant les valeurs d'attribut pour l'entité gérée equipmentHolderFManagedEntity	
EquipmentType	Cet élément de données désigne l'item d'inventaire nommé dans l'évènement d'autodécouverte.	short	
EthernetPMHistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
EthernetProfileType	Cet élément de données donne les valeurs pour un profil de sorte Ethernet.	struct	
ExternalTimeType	Cet élément de données établit l'heure locale qu'il faut associer à la ressource du réseau.	GeneralizedTimeType	
FaxDemodIndType	Cet élément de données booléen indique si une démodulation FAX est présente, la valeur TRUE impliquant sa présence.	boolean	
FilenameType	Désigne le nom du chemin d'accès complet d'un fichier. Ce fichier peut contenir des données de configuration téléexportées vers un élément de réseau, une destination de transfert d'un fichier d'articles, ou un emplacement pour un logiciel (générique ou correctif) d'élément de réseau.	string	
FileTransferHistorySeqType	Cet élément de données donne tous les articles enregistrés concernant des transferts de fichiers encore journalisés sur le système de gestion par fournisseur.	struct	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`FilterType`	Cet élément de données est défini par CosNotifyFilter::Filter.		
`FMDataIndType`	Cet élément de données booléen indique si des données en mode trame sont transportées sur cette connexion, la valeur TRUE impliquant sa présence.	`boolean`	
`FMMaxFrameLenType`	Cet élément de données indique la longueur maximale d'une unité de données en mode trame.	`long long`	
`ForwardDelayType`	Cet élément de données indique la durée (en centièmes de seconde) pendant laquelle le pont situé sur la carte Ethernet dans la terminaison ONT (en tant que membre de la communauté de tous les ponts contenus dans le réseau de zone locale ponté) conserve un paquet avant de le réexpédier.	`short`	
`ForwardErrorCorrectionType`	Cet élément de données indique la méthode de correction CED.	`enum`	
`FullActionType`	Spécifie le comportement de la chaîne d'articles quand elle atteint sa longueur maximale.	`enum`	Choix entre débordement et arrêt.
`GeneralizedTimeType`	Fournit une mesure de temps normalisée. Utilisé afin d'éviter des ambiguïtés mettant en jeu des fuseaux horaires.	`string`	Un seul format est autorisé: YYYYMMDDHHMMSS.fffZ (c'est-à-dire GMT).
`HelloTimeType`	Cet élément de données indique la durée (en centièmes de seconde) entre paquets de préappel. C'est la durée, en centièmes de seconde, pendant laquelle un pont signale sa présence en tant que racine existante ou potentielle.	`short`	
`HistoryDataType`	Désigne un type d'enregistrement contenant des statistiques de surveillance de la qualité de fonctionnement.	`RecordKindType`	
`HourlyDailyWeeklyMonthlyIndType`	Spécifie si le programme de planification a un cycle temporel de toutes les heures, tous les jours, toutes les semaines ou tous les mois.	`enum`	Toutes les heures, tous les jours, toutes les semaines, tous les mois

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
IDLCCallProcessing ProfileType	Cet élément de données donne les valeurs d'un profil de sorte: traitement d'appel par circuit IDLC.	struct	
JitterBufferMaxType	Cet élément de données donne la capacité maximale du tampon de gigue associé à ce service. Les unités sont en millisecondes.	long long	
JitterTargetType	Cet élément de données donne la valeur cible du tampon de gigue. Le système essaiera de conserver le tampon de gigue à la valeur cible. Les unités sont en millisecondes.	long long	
LANType	Cet élément de données donne des informations sur le type de LAN employé, par exemple, Ethernet, à jetons, etc.	short	
LESProfileType	Cet élément de données donne les valeurs pour un profil de sorte LES.	struct	
LocalMaxNumVCC SupportedType	Cet élément de données désigne le nombre de connexions VCC qui peuvent être prises en charge par l'élément de réseau en mode ATM à cette extrémité de l'interface.	long long	
LocalMaxNumVCIBitsType	Cet élément de données désigne le nombre maximal de bits attribués du sous-champ d'identificateur VCI qui peuvent être pris en charge par l'élément de réseau FSAN à cette extrémité de l'interface.	short	
LocalMaxNumVPC SupportedType	Cet élément de données désigne le nombre de connexions VPC qui peuvent être prises en charge par la terminaison OLT à cette extrémité de l'interface.	long long	
LocalMaxNumVPIBitsType	Cet élément de données désigne le nombre maximal de bits attribués au sous-champ d'identificateur VPI qui peuvent être pris en charge par l'élément de réseau FSAN à cette extrémité de l'interface.	short	
LoopbackCodeType	Cet élément de données désigne le type de code de rebouclage pris en charge dans la bande.	enum	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
LoopbackInfoType	Cet élément de données spécifie le type, le sens et l'emplacement du rebouclage.	struct	
LoopbackInfoSeqType	Fournit une énumération de toutes les conditions de rebouclage dans les ressources de réseau.	struct	
LoopbackLocationIdSeqType	Spécifie l'identificateur d'emplacement unique pour une extrémité ctp.	Séquence d'octets de longueur 16	
LoopbackLocCodeType	Cet élément de données donne le code qui désigne les cellules OAM de rebouclage entrantes en mode ATM qui doivent être rebouclées vers cette interface UNI.	string	
LoopbackTestType	Cet élément de données identifie le type de montage de test de bouclage à utiliser ou en cours d'utilisation.	unsigned short	Valeurs possibles définies dans l'interface de bouclage PhysicalLayer de Q.834::Common.
LoopbackTrackingObjectIdType	Cet élément de données identifie l'état du bouclage en cours.	TrackingObjectIdType	
MACBridgePortPMHistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
MACBridgePMHistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
MACBridgeServiceProfileType	Cet élément de données donne les valeurs pour un profil de sorte MACBridgeService.	struct	
ManagedEntityIdType	Cet élément de données donne un nom unique pour une entité gérée. Ce nom indique s'il représente un objet géré à granulométrie fine, un objet de façade, ou juste une référence à une structure de données.	struct de: enum et RDNType	
MaxAgeType	Cet élément de données indique la durée de vie maximale (en secondes) pour une entrée dans la liste de l'arbre de recouvrement maximal. Il indique la durée de vie maximale en secondes des informations protocolaires reçues avant leur rejet.	short	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`MaxBSEgressType`	Longueur maximale de rafale – ce paramètre est requis pour le trafic VBR en temps réel et en temps planifié ainsi que pour le trafic à débit GFR. Il s'applique aux flux de trafic à priorité CLP = 0 + 1 aux débits VBR.1, GFR.1 et GFR.2, et s'applique aux flux de trafic à priorité CLP = 0 aux débits VBR.2 et VBR.3.	`long long`	
`MaxBSIngressType`	Longueur maximale de rafale – ce paramètre est requis pour le trafic à débit VBR en temps réel et en temps planifié ainsi que pour le trafic à débit GFR. Il s'applique aux flux de trafic à priorité CLP = 0 + 1 aux débits VBR.1, GFR.1 et GFR.2, et s'applique aux flux de trafic à priorité CLP = 0 aux débits VBR.2 et VBR.3.	`long long`	
`MaxCPCSSDUSizeType`	Cet élément de données à valeurs multiples représente la longueur maximale des unités CPCS_SDU qui sera transmise sur la connexion dans les deux sens de transmission: entrant (vers l'avant) et sortant (vers l'arrière).	`struct`	
`MaxCPS_SDULengthType`	Cet élément de données donne la longueur maximale autorisée de l'unité de données de service de sous-couche de convergence de partie commune (ou SDU de CPS) qui sera autorisée sur la connexion dans le sens de transmission amont ou aval.	`long long`	
`MaxFrameSizeType`	Cet élément de données indique la longueur maximale autorisée de trame pouvant être transmise de part et d'autre de cette interface.	`long long`	
`MaximumChanIdType`	Cet élément de données donne la valeur maximale d'identificateur de voie autorisée pour la voie dans la connexion.	`short`	
`MaxNumChannelsType`	Cet élément de données donne le nombre maximal de voies qui peuvent être acheminées par chemin de voie virtuelle associé à l'extrémité vcCTP d'interfonctionnement.	`short`	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`MaxNumCIDsType`	Cet élément de données spécifie le nombre maximal de voies qui peuvent être actives dans la connexion VCC.	`short`	
`MaxPacketLengthType`	Cet élément de données spécifie la longueur maximale de paquet.	`long long`	
`MaxSizeType`	Spécifie la longueur maximale de la chaîne d'articles.	`unsigned long long`	En unités à déterminer par accord mutuel entre fournisseur et opérateur.
`MaxSSSARSDULengthType`	Cet élément de données donne la longueur maximale autorisée pour une unité SSSAR-SDU de la sous-couche de convergence spécifique du service de segmentation et réassemblage.	`long long`	
`MFR1IndType`	Cet élément de données booléen indique si les chiffres de numérotation multifréquence R1 sont transportés sur la connexion, la valeur TRUE impliquant sa présence.	`boolean`	
`MFR2IndType`	Cet élément de données booléen indique si les chiffres de numérotation multifréquence R2 sont transportés sur la connexion, la valeur TRUE impliquant sa présence.	`boolean`	
`MFSEgressType`	Longueur maximale de trame – ce paramètre n'est requis que pour le trafic à débit GFR.	`long long`	
`MFSIngressType`	Longueur maximale de trame – ce paramètre n'est requis que pour le trafic à débit GFR.	`long long`	
`MinimumChanIdType`	Cet élément de données donne la valeur minimale d'identificateur de voie autorisée pour toute voie dans la connexion.	`short`	
`MonitoredParameterSeqType`	Cet élément de données énumère des paramètres de qualité de fonctionnement.	`string`	Paramètres surveillés définis dans `Q834Common::MonitoredParameter`
`MonitoringKindType`	Cet élément de données identifie le type de surveillance de performance spécifié.	`string`	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`MonitoringPoint ThresholdsType`	Cet élément de données énumère des instances de point de surveillance et leurs données associées de seuil pour une ressource réseau spécifiée.	Séquence de: `struct`	
`NameType`	Cet élément de données donne un nom CORBA pour un objet de profil	`CosNaming:: Name`	
`NEFSANType`	Cette structure de données désigne l'item d'inventaire d'équipements génériques nommé dans l'évènement d'autodécouverte.	`struct`, énumérant les valeurs d'attribut pour l'entité gérée `NEFSAN`.	
`NEKindType`	Cet élément de données désigne le type des éléments de réseau qui peuvent être construits.	`enum`	Le choix est entre `BPONOLT`, `BPONONT`, `BPONONU`, et `BPONNT` (Note).
`NotificationIdentifier Type`	Désigne de façon unique la notification.	`long long`	
`NTType`	Cette structure de données désigne l'item d'inventaire de terminaisons NT nommé dans l'évènement d'autodécouverte.	`struct`, énumérant les valeurs d'attribut pour l'entité gérée de terminaison `NT`.	
`OLTType`	Cette structure de données désigne l'item d'inventaire de terminaisons OLT nommé dans l'évènement d'autodécouverte.	`struct`, énumérant les valeurs d'attribut pour l'entité gérée de terminaison `OLT`.	
`ONTType`	Cette structure de données désigne l'item d'inventaire de terminaisons ONT nommé dans l'évènement d'autodécouverte.	`struct`, énumérant les valeurs d'attribut pour l'entité gérée de terminaison `ONT`.	
`ONUType`	Cette structure de données désigne l'item d'inventaire de terminaisons ONU nommé dans l'évènement d'autodécouverte.	`struct`, énumérant les valeurs d'attribut pour l'entité gérée de terminaison `ONU`.	
`OperationalStateType`	Specifie l'état opérationnel (activé ou désactivé) de l'entité gérée.	`enum`	Défini dans la Rec. UIT-T X.780.

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
OpticalWaveLengthArraySeqType	Spécifie la longueur d'onde (en nanomètres) et le sens de chaque longueur d'onde multiplexée à cet accès optique.	struct	
ParameterSettingSeqType	Cet élément de données énumère des paramètres de surveillance en même temps que leurs paramètres de fenêtre mobile associés.	struct	
PartiallyFilledCellsType	Cet élément de données booléen désigne le nombre d'octets initiaux en usage.	long long	
PasswordType	Offre un justificatif de sécurité pour l'accès à l'application ou aux serveurs du système de gestion par fournisseur.	string	Un mot de passe fourni avec un identifiant d'utilisateur doit toujours se conformer à la politique de mots de passe mise en œuvre par le système de gestion du fournisseur.
PasswordPolicyType	Spécifie la politique relative aux mots de passe qui est mise en œuvre par le système de gestion par fournisseur. Il est composé de deux éléments: UserLoginPolicyType et SessionPolicyType.	struct	
PCMEncTypeType	Cet élément de données indique le type de codage MIC. Les valeurs valides comprennent sans y être limitées: "codage en loi mu" et "codage en loi alpha".	short	
PCREgressType	Débit cellulaire crête – ce paramètre est requis pour le trafic de toutes les catégories de service. Il s'applique aux flux à priorité CLP = 0 au débit ABR et sinon aux flux à priorité CLP = 0 + 1.	long long	
PCRIngressType	Débit cellulaire crête – ce paramètre est requis pour le trafic de toutes les catégories de service. Il s'applique aux flux à priorité CLP = 0 au débit ABR et sinon aux flux à priorité CLP = 0 + 1.	long long	
PerceivedSeverityType	Défini par la Rec. UIT-T X.780.	enum	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`PIDType`	Cet élément de données désigne les valeurs de type de média qui peuvent être utilisées dans l'encapsulation ATM (définie dans RFC 1483).	`short`	
`PlugInUnitFType`	Cette structure de données désigne l'item d'inventaire PlugInUnitF nommé dans l'évènement d'autodécouverte.	`struct`, énumérant les valeurs d'attribut pour l'entité gérée de type de module d'extension.	
`PlugInUnitType`	Propre à l'implémentation, nom indiqué par le fournisseur de la carte d'équipement de ligne.	`string`	
`PortBandwidthSeqType`	Cet élément de données énumère les largeurs de bande par accès préfourni.	`struct`	Exemples: `ReservedBandwidthSeqType` et `AvailableSysBandwidthSeqType`
`POTSSignallingType`	Cet élément de données sélectionne le format de signalisation qui devrait être utilisé pour le service RTC.	`enum`	
`ProbableCauseType`	Données de cause probable.	`unsigned short`	Valeurs: définies dans `Q834Common::ProbableCause` et Rec. UIT-T X.780
`ProceduralStatusSetType`	Fournit l'état procédural d'une activité non terminée.	Séquence de: `enum`	Défini dans la Rec. UIT-T X.780.
`ProfileInfoType`	Cet élément de données désigne le type de profil et ses valeurs d'attribut.	`struct`	
`ProfileKindType`	Cet élément de données désigne le type de profil nommé.	`unsigned short`	Valeurs fournies dans `Q834Common`
`ProtectionParameterType`	Cet élément de données décrit les paramètres de commutation sur secours associés à un groupement de protection d'entités gérées. Ces paramètres comprennent taux de commutations sur secours, mécanismes autorisés de commutation sur secours, indicateur d'inversion automatique et temps d'attente jusqu'à l'inversion automatique.	`struct`	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
ProtectionUnitType	Cet élément de données associe les ressources de réseau protégées et protégeantes.	struct	
RASTimerType	Cet élément de données indique le temps de réassemblage (en secondes) de la sous-couche de convergence spécifique du service de segmentation et réassemblage pour I.366.1.	short	
RateControlIndType	Cet élément de données booléen indique si la commande de débit est transportée ou non sur la connexion, la valeur TRUE impliquant sa présence.	boolean	
RecordType	Il s'agit d'un item mémorisé dans une chaîne d'articles.	any	La valeur est une structure fondée sur le type RecordKindType défini dans Q834Common::RecordSetType. Un seul type de sorte d'article peut être mémorisé dans une chaîne d'articles.
RecordSetIdType	Identificateur d'entité gérée de la chaîne d'articles.	ManagedEntityIdType	
RecordSetStatusType	Spécifie l'état actuel de la chaîne d'articles.	struct de: currentSizeType, OperationalStateType, MaxSizeType, SizeThresholdType, filterName, Administra-tiveStateType, RecordKindType et RecordSetUserLabel	
RecordKindType	Désigne le type de l'article journalisé.	unsigned short	Valeurs définies dans Q834Common::RecordSetType
RecordsSeqType	Cet élément de données donne un ensemble d'articles regroupés par type.	Séquence de: any	Voir ci-dessus.
RemainingPasswordValidity	Spécifie la validité du mot de passe en nombre de jours.	long	Valeur ≥ 1

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`ReservationIdType`	Identificateur qui associe la largeur de bande réservée sur un réseau PON ou dans un système de terminaison OLT avec une instance de service.	`string`	
`ReservationInfoType`	Cet élément de données donne un état complet des informations de réservation et des informations de connexion de service associées.	`struct`	
`ReservedBandwidthSeq Type`	Quantifie la grandeur de la largeur de bande associée à un identificateur de réservation.	Séquence de: `PortBandwidth Type`	
`ResourceSelfTestInfo Type`	Fournit la sélection des diagnostics d'essai à utiliser dans l'essai d'autovérification d'une ressource recherchée.	`short`	Valeurs déterminées par réalisation du fournisseur.
`ResourceSelfTestResult Type`	Spécifie le résultat pour l'essai d'autovérification d'une ressource.	`struct`	
`ScheduledFileTransfer Type`	Enumère toutes les activités planifiées de transfert de fichiers en instance pour le système de gestion par fournisseur.	`struct`	
`ScheduledSynchNEType`	Cet élément de données énumère toutes les activités planifiées de synchronisation d'éléments de réseau pour le système de gestion par fournisseur.	`struct`	
`ScheduledTestNESeqType`	Cet élément de données énumère toutes les activités planifiées d'essais pour le système de gestion par fournisseur.	`struct`	
`SchedulerType`	Fournit toutes les instanciations des programmateurs de planification dans le système de gestion par fournisseur.	`struct`	
`SCREgressType`	Débit cellulaire soutenable – ce paramètre s'applique au débit VBR en temps réel et en temps planifié. Il s'applique aux flux de trafic à priorité CLP = 0 + 1 aux débits VBR.1 et s'applique aux flux de trafic à priorité CLP = 0 aux débits VBR.2 et VBR.3.	`long long`	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
SCRIngressType	Débit cellulaire soutenable – ce paramètre s'applique au débit VBR en temps réel et en temps planifié. Il s'applique aux flux de trafic à priorité CLP = 0 + 1 aux débits VBR.1 et s'applique aux flux de trafic à priorité CLP = 0 aux débits VBR.2 et VBR.3.	long long	
SecurityAlarmLogRecordType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
SegmentEndpointIndType	Indique si l'extrémité vpNetworkCTP construite doit être considérée comme un segment ou comme une extrémité pour les essais de rebouclage de flux de cellules OAM en mode ATM.	enum	choix de segment, d'extrémité, ou néant
SegmentLengthType	Cet élément de données donne la longueur de segment pour la sous-couche de convergence propre au service de segmentation et réasssemblage. Il s'étend de 0 à la valeur maximale fournie par l'élément de données MaxCPS_SDULen.	long long	
SerialNumType	Fournit l'identificateur unique de matériel utilisé dans l'activité de télémétrie définie dans la Rec. UIT-T Q.983.1.	string	
ServiceAffectingType	Cet élément de données indique si une condition d'échec affecte actuellement le service et si elle peut être déterminée.	enum	
ServiceCategoryType	Indique la catégorie de service telle que définie dans ATM-Forum TM 4.0. Les valeurs valides sont: CBR, rt-VBR, nrt-VBR, UBR, ABR ou GFR.	enum	
ServiceCatType	Cet élément de données indique le type de catégorie de service fourni par la couche AAL2. Les valeurs valides comprennent sans y être limitées: "Audio" et "Multirate".	enum	
ServiceInstanceIdType	Etiquette fournie par l'opérateur et associée à des connexions par le système de gestion du fournisseur.	string	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
ServiceOutageRecordType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
SessionPolicyType	Spécifie des règles régissant des sessions de client d'interface GUI interconnectées au système de gestion par fournisseur.	struct	
SizeThresholdType	Spécifie le seuil dans la chaîne d'articles afin de déclencher des alarmes.	unsigned short	La valeur est interprétée comme un pourcentage en nombre entier (0-100%) de MaxSize
SlotAssignmentType	Fournit la liste des logements préfournis dans le nœud.	Séquence de: paires, chaque paire montrant le numéro de logement et le type de module d'extension.	
SoftwareDownloadTrackingObjectIdType	Fournit une liste complète des activités de téléchargement de logiciels qui sont en cours ou n'ont pas réussi à s'effectuer.	TrackObjectIdType	
SONETSDHLinePMHistoryData	Fournit la structure de l'article journalisé détaillant les données de qualité de fonctionnement recueillies dans un intervalle de 15 min pour le point de surveillance rsTTPF.	struct	
SONETSDHSectionAdaptationPMHistoryData	Fournit la structure de l'article journalisé détaillant les données de qualité de fonctionnement recueillies dans un intervalle de 15 min pour le point de surveillance au3CTPF ou au4CTPF.	struct	
SONETSDHSectionPathPMHistoryData	Fournit la structure de l'article journalisé détaillant les données de qualité de fonctionnement recueillies dans un intervalle de 15 min pour le point de surveillance msTTPF, vc3CTPF ou vc4CTPF.		

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
SpanningTreeIndType	Cet élément de données booléen indique si un algorithme d'arbre de recouvrement maximal est activé. La valeur TRUE signifie l'activation.	boolean	
SpecificProblems	Cet élément de données énumère des valeurs de code pour tous les problèmes spécifiques associés à une condition d'échec et d'alarme.	Séquence de: long	Valeurs définies par fournisseur.
SSADTIndType	Cet élément de données booléen indique si le mécanisme de transfert assuré de données a été sélectionné, la valeur TRUE indiquant la sélection.	boolean	
SSCSParameterProfile1 Type	Cet élément de données donne les valeurs pour un profil de sorte Profil 1 de paramètre SSCS.	struct	
SSCSParameterProfile2 Type	Cet élément de données donne les valeurs pour un profil de sorte Profil 2 de paramètre SSCS.	struct	
SSCSTypeType	Cet élément de données désigne le type de sous-couche SSCS pour la couche AAL.	enum	
SSTEDIndType	Cet élément de données booléen indique si les mécanismes de détection d'erreur de transmission ont été sélectionnés, la valeur TRUE indiquant la sélection.	boolean	
StateChangeDefinition Type	Cet élément de données énumère toutes les modifications de variable d'état et de statut déclenchées par une condition d'échec provoquant une alarme.	AttributeChan geSetType	Importés de la Rec. UIT-T X.780.
StatusAttributeType	Fournit une énumération montrant le type StatusValueType, et le pourcentage d'achèvement pour la sauvegarde ou la restauration de données de configuration associées à un élément de réseau.	struct	
StatusValueType	Fournit un choix de valeurs entre la valeur ProceduralStatusSetType et d'autres valeurs de statut indiquant des aspects relatifs à l'achèvement.	union	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
`StructuredDataTransferType`	Cet élément de données booléen indique si le transfert de données structurées (SDT) a été configuré dans la couche AAL.	`boolean`	
`SubType`	Cet élément de données désigne le sous-type de couche AAL.	`enum`	
`SupplyPowerIndType`	Cet élément de données indique si l'accès associé est une source d'alimentation.	`boolean`	
`SWPValueType`	Fournit une énumération de points de surveillance avec leurs paramètres de fenêtre mobile associés.	`struct`	
`SynchChangeIndType`	Cet élément de données booléen indique si la synchronisation des changements dans le fonctionnement de la sous-couche SSCS est transportée par la connexion, la valeur TRUE impliquant sa présence.	`boolean`	
`SystemTimingType`	Spécifie les horloges de référence primaire et secondaire pour l'élément de réseau.	`struct de: struct`	
`T303Type`	Cet élément de données définit la durée maximale en millisecondes pendant laquelle la terminaison ONT attendra une réponse au message "SETUP" message trouvé en couche 3 pour le TMC ou le CSC.	`enum`	
`T396Type`	Cet élément de données spécifie la durée pendant laquelle la terminaison ONT attendra une réponse à un message "SETUP" après l'expiration initiale de la temporisation T303.	`enum`	
`TargetType`	Cet élément de données énumère les ressources de réseau vers lesquelles des logiciels doivent être acheminés pour distribution régie par le système de gestion du fournisseur.	Séquence de: `struct` de `ManagedEntityIdType` et `string`	Permet la sélection au niveau d'un système, d'un élément de réseau, d'un type de module d'extension, ou d'un logement spécifique.

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
TargetActivityType	Fournit l'énumération des niveaux d'activité, des types d'activité et des domaines administratifs à associer à un utilisateur ou à un groupe d'utilisateurs.	struct	
TCAdaptionProtocol MonitoringPMHistory Data	Fournit la structure de l'article journalisé détaillant les données de qualité de fonctionnement recueillies dans un intervalle de 15 min pour points de surveillance de tcAdaptorF.		
TestHistoryType	Cet élément de données énumère les informations archivées sur une activité d'essai effectuée pour le système de gestion par fournisseur.	struct	
TestIterationNumType	Cet élément de données spécifie le nombre de fois qu'un test doit être répété.	short	
TestTrackingObjectId Type	Cet élément de données identifie un test en cours ou en instance dans le système de gestion par fournisseur.	Tracking ObjectId	
ThresholdDataProfile Type	Cet élément de données est une énumération de paires: une instance thresholdDataProfile et un type de point de surveillance.	séquence de: struct	
ThresholdDataType	Cet élément de données énumère les noms de paramètre de qualité de fonctionnement avec une valeur de seuil pour chacun.	struct	
ThresholdInfoType	Cet élément de données fournit des informations sur un évènement d'alarme de qualité de service et indique l'identité du paramètre de qualité de fonctionnement, sa valeur observée, et les valeurs supérieure et inférieure qui définissent son étendue de valeurs de seuil.	struct	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
ThresholdsList	Cet élément de données énumère les relations entre les types de terminaison de surveillance et leur profil de données de seuil associé.	Séquence de: struct, le premier composant indiquant le type de point de surveillance et le second indiquant la référence à un profil de données de seuil.	
ThresholdsType	Cet élément de données énumère les types de point de surveillance associés à leur nom de profil de données de seuil.	struct	
TimerCULengthType	Cet élément de données donne la valeur de la temporisation d'"utilisation combinée": Timer_CU.	long long	
TimingReferenceType	Cet élément de données définit la façon dont le rythme interne est calculé.	enum	
TotalEgressBandwidthType	Cet élément de données désigne la grandeur totale de la largeur de bande sortante pour une interface ATM.	long long	
TotalIngressBandwidthType	Cet élément de données désigne la grandeur totale de la largeur de bande entrante pour une interface ATM.	long long	
TrackingObjectIdType	Cet élément de données aide à identifier l'état d'une activité requise dont la durée d'achèvement est plus longue que quelques secondes.	unsigned long	Exemples: TransferTrackingObjectIdType, SoftwareDownloadTrackingObjectType, et testTrackingObjectIdType
TrafficDescriptorProfileType	Cet élément de données donne les valeurs pour un profil de sorte Traffic Descriptor.	struct	
TransferTrackingObjectIdType	Fournit une liste de toutes les activités de téléexportation vers une base MIB, en instance et planifiées dans le système de gestion par fournisseur.	string	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
TriggerTimeMatrixType	Spécifie un instant de déclenchement.	struct	
UNIProfileType	Cet élément de données donne les valeurs pour un profil de sorte UNI.	struct	
UPCNPCDisagreementPM HistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
UPCNPCIndicatorType	Cet élément de données booléen détermine si la surveillance est effectuée pour toutes les connexions à l'interface.	boolean	
UsageStateType	Cet élément de données énumère les valeurs pour la variable d'état d'utilisation.	enum	Les valeurs valides d'énumération sont: idle, active, busy. Définies dans la Rec. UIT-T X.780.
UserGroupId	Il s'agit du nom fourni au système de gestion par fournisseur par l'opérateur pour un groupement d'utilisateurs.	UserLabelType	Ne peut être la chaîne vide.
UserGroupType	Fournit l'étiquette d'utilisateur pour le groupe d'utilisateurs, énumère des utilisateurs qui sont membres du groupe, et énumère leurs activités cibles.	struct de: UserGroupId Type, UserIdSeq Type, et TargetActi vitySeqType	
UserIdType	Etiquette d'utilisateur attribuée à des utilisateurs du système de gestion par fournisseur.	UserLabel	
UserLabelType	Fournit un identificateur qui est créé et fourni par l'opérateur ou par le système OMS, à associer à une ressource gérée par le système de gestion du fournisseur.	string	
UserLoginPolicyType	Spécifie la politique régissant l'identifiant de l'utilisateur.	struct	
UserLoginPolicy ViolationReasonType	Spécifie les raisons du rejet de l'attribution d'un mot de passe à un utilisateur.	enum	
UserType	Fournit l'identifiant d'utilisateur, les groupes d'utilisateurs auxquels l'utilisateur appartient et les activités cibles attribuées à l'utilisateur.	struct de: UserIdType, UserGroupId SeqType, et Target ActivitySeq Type	

Tableau A.1/Q.834.4 – Eléments de données et définitions

Nom d'élément de données	Définition	Syntaxe	Commentaires
VersionType	Fournit l'identifiant de la version du matériel ou du logiciel pour le système par type d'identificateur d'entité gérée	struct	
VoicePMHistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
VoiceServicesProfile AAL1Type	Cet élément de données donne les valeurs pour un profil de sorte services téléphoniques, au moyen de la couche AAL1.	struct	
VoiceServicesProfile AAL2ProfileType	Cet élément de données donne les valeurs pour un profil de sorte services téléphoniques au moyen de la couche AAL2.	struct	
VPVCPMHistoryDataType	Cet élément de données indique les items contenus dans le présent type d'article journalisé.	struct	
NOTE – Si l'on envisage d'utiliser le module Q834Build dans la gestion d'autres types de technologie d'accès, il faut soit développer les valeurs énumérées, soit changer la syntaxe en "court", avec des constantes qui identifient les types d'élément de réseau à construire.			

Annexe B

Exceptions

Le Tableau B.1 donne la liste de toutes les exceptions possibles et des circonstances dans lesquelles elles peuvent être déclenchées par une ou plusieurs opérations dans la présente spécification d'interface.

Tableau B.1/Q.834.4 – Exceptions

Exception déclenchées	Description
AccessDenied	Le système n'est pas autorisé à accéder à cet objet d'interface.
ActivationCompleted	Indique quelle activation de logiciel a été exécutée de façon que l'activation ne puisse pas être annulée.
ActivationFailure	Echec du processus d'activation de logiciel.
ActivityCompleted	L'activité logicielle a été exécutée et ne peut pas être annulée.
ActivityInProgress	Cette exception est déclenchée quand l'activité logicielle a été lancée et ne peut pas être annulée.
AddressLabelMismatch	L'élément de réseau désigné ne possède pas l'adresse actuelle RCD fournie dans la demande.

Tableau B.1/Q.834.4 – Exceptions

Exception déclenchées	Description
APONLayerFailure	Il y a eu une panne de télémétrie par protocole de réseau APON entre la terminaison OLT et le nœud sous-jacent désigné.
BackupInProgress	Cette exception est déclenchée quand la demande est émise pendant que la sauvegarde est en cours.
CannotAssignManagedEntityId	Le système de gestion par fournisseur n'a pas été en mesure de régler l'identificateur d'entité gérée pour la terminaison OLT, indiquant ainsi que le système de gestion par fournisseur n'est pas en mesure de gérer la terminaison OLT.
CannotRetrieveUserLabel	Le système de gestion par fournisseur n'a pas été en mesure de lire l'étiquette d'utilisateur préfournie sur la terminaison OLT.
CollectionLimitation	Le système de gestion par fournisseur ne peut pas acquérir de données pendant l'intervalle temporel et la période de granularité impartis, en raison de restrictions d'implémentation.
CollectionPeriodPast	Exception indiquant que l'instant de fin de période est inférieur ou égal à l'heure actuelle.
CommFailure	Il y a eu une panne de liaison RCD entre l'élément de réseau et le système de gestion par fournisseur.
ConnectionAlreadyExists	Il existe déjà une connexion de sous-réseau avec les mêmes extrémités.
ConnectionCountExceeded	Le nombre maximal de connexions pour la terminaison OLT ou l'accès de réseau PON a été dépassé par cette demande de fourniture de services.
DCNTimeout	La liaison de communications RCD entre au moins un des éléments de réseau et le système de gestion par fournisseur est encombrée au point que les informations d'état actuel ou de statut ne peuvent pas être transférées dans un intervalle de synchronisation défini par le système.
DeniedAccess	Le système n'est pas autorisé à accéder à l'élément de réseau.
DuplicateProfileName	Cette exception est déclenchée lorsque le nouveau nom de profil copie un nom de profil existant.
DuplicateSerialNumber	Il existe un autre équipement du même type avec ce numéro de série.
DuplicateUserGroupId	L'identificateur est déjà utilisé pour un autre groupe d'utilisateurs.
DuplicateUserId	Le profil de contrôle d'accès a déjà été établi pour cet identifiant d'utilisateur.
DuplicateUserLabel	L'étiquette d'utilisateur fournie dans la demande a servi à étiqueter une autre entité gérée au même instant.
EquipmentFailure	L'élément de réseau a actuellement une condition d'échec empêchant l'exécution de la transaction requise.
HWServicesMismatch	L'élément de remplacement ne peut pas exécuter les services préfournis.
InstallationFailure	Echec du processus d'installation de logiciel.

Tableau B.1/Q.834.4 – Exceptions

Exception déclenchées	Description
InsufficientBW	L'algorithme de commande CAC indique que le service requis a besoin d'une trop grande partie de la largeur de bande pour la terminaison OLT.
InsufficientMemory	Il n'y a pas assez de mémoire dans l'élément de réseau pour charger l'unité logicielle.
InsufficientPONBW	La terminaison ONT ou ONU ne peut pas être télémesurée en raison d'une largeur de bande insuffisante sur la liaison de réseau APON.
InterfaceSpeedNotChangeable	L'accès physique ne peut pas prendre en charge la nouvelle vitesse d'interface ou la vitesse n'est pas configurable.
IntervalCountTooLarge	Cette exception est déclenchée quand les intervalles requis dépassent le nombre maximal pris en charge par le système de gestion par fournisseur. Cette exception indique le nombre maximal autorisé d'intervalles de surveillance pris en charge par le système de gestion par fournisseur.
InvalidDCNAddress	L'adresse RCD spécifiée n'est pas valide.
InvalidEquipmentCode	Le code d'équipement n'est pas conforme à la syntaxe.
InvalidExternalTime	Le temps externe spécifié n'est pas valide.
InvalidLocationId	L'identificateur LLID spécifié n'est pas valide.
InvalidPort	L'accès PON spécifié n'est pas valide.
InvalidProtectionScheme	La ressource réseau ne prend pas en charge les paramètres de protection spécifiés dans le contexte avec l'énumération des accès ou les unités de protection sont des accès ayant des caractéristiques de trajet physique dissemblables.
InvalidScheduler	Les valeurs paramétriques du programmateur sont en dehors du domaine d'application défini.
InvalidSerialNumSyntax	La syntaxe du numéro de série fourni ne concorde pas avec les règles de définition.
InvalidSlotAssignmentList	Les règles de fourniture de logement prévues sont violées par l'attribution de logement fournie.
InvalidSoftwareTracking Object	L'objet de suivi logiciel mentionné n'est pas le plus récent associé à l'installation d'un logiciel dans l'élément de réseau.
InvalidStartTime	L'instant de début est incompatible avec l'instant du moment, le déclencheur temporel du moment ou le nouvel instant d'arrêt.
InvalidStopTime	Le nouvel instant d'arrêt est incompatible avec la matrice actuelle des déclencheurs temporels, avec l'instant du moment ou avec le nouvel instant de début.
InvalidTestOperations	L'opération d'autovérification requise n'est pas valide.
InvalidTimeoutPeriod	La période de temporisation indiquée viole la définition de valeurs valides.
InvalidTrigger	Le déclencheur spécifié a des valeurs qui ne peuvent pas être interprétées par le programmateur de planification.
InvalidUserLabelSyntax	L'étiquette d'utilisateur ne suit pas les règles établies pour l'étiquette d'utilisateur.

Tableau B.1/Q.834.4 – Exceptions

Exception déclenchées	Description
LockedAlready	L'état administratif de l'entité gérée nommée est déjà bloqué et il n'a pas exécuté sa fonction normale.
MaxSubtendingNodesExceeded	Le nombre maximal configuré de nœuds sous-jacents pour l'interface PON désignée a été dépassé par cette demande de fourniture ou de réservation de services.
NoResponse	La terminaison ONT ou ONU n'a pas pu être télémesurée et cet échec est dû à une autre cause qu'un problème détecté concernant la syntaxe du numéro de série, le protocole de couche APON, ou des étiquettes d'utilisateur en double ou non valides.
NoSuchRecords	Aucun article parmi les chaînes d'articles désignées ne correspond aux critères de sélection.
NoSynchInProgress	L'exception est déclenchée s'il n'y a pas de processus de synchronisation en cours.
NotAvailableForTest	La terminaison CTP spécifiée n'est pas disponible pour cet essai.
ParameterViolation	Cette exception est déclenchée quand les paramètres d'extrémité ne correspondent pas aux caractéristiques protocolaires de l'accès, ou quand les valeurs sont hors étendue ou sont des copies non valides.
ProfileInUse	Cette exception est déclenchée lorsque le profil ne peut pas être supprimé parce qu'il est encore utilisé pour caractériser des entités gérées dans la juridiction de gestion du système de gestion par fournisseur.
ProfileSuspended	Le ou les profils nommés dans l'invocation ont été suspendus pour utilisation par le système OMS ou par l'opérateur dans le cadre du système de gestion par fournisseur.
RecordSetExists	La chaîne d'articles définie par les paramètres de la demande de création existe déjà dans le système de gestion par fournisseur.
RemainingContainedManaged Entities	La carte d'équipement de lignes ou les logements d'équipement confinés n'ont pas encore été supprimés.
RemainingReservations	Le nœud ne peut pas être supprimé car des réservations de ressources existent encore.
RemainingSubnetwork Connections	Le nœud du module d'extension ne peut pas être supprimé car des connexions de sous-réseau existent encore.
ScheduleInUse	Il reste des activités planifiées par le programmateur nommé.
SlotAlreadyAssigned	L'alvéole requis est déjà préfourni.
SoftwareLoadHardwareMismatch	Les précédentes données de configuration d'élément de réseau n'ont pas pu être téléimportées par l'élément de réseau parce que les modifications apportées au matériel d'élément de réseau ont provoqué une incapacité.
SoftwareLoadHWMismatch	Le logiciel désigné ne peut pas être chargé sur le matériel de l'équipement étant donné que la version du matériel ne peut pas accepter la charge logicielle.
SoftwareNotYetInstalled	Le logiciel ne peut pas être activé étant donné qu'il n'a pas encore été installé.
SoftwareTrackingObjectInUse	Des activités logicielles en souffrance sont suivies par cet objet et celui-ci ne peut donc pas être supprimé.

Tableau B.1/Q.834.4 – Exceptions

Exception déclenchées	Description
SourceUnreachable	Le serveur détenant la charge logicielle à télécharger n'a pas pu être atteint par la terminaison OLT.
SynchNotScheduled	Cette exception est déclenchée lorsque le programme de modification désigne un élément de réseau pour lequel n'a pas été établi précédemment de programme de synchronisation.
Timeout	La durée du processus a atteint une limite de temporisation définie par le système avant que le processus puisse s'achever.
TooManyNEs	Le système de gestion par fournisseur ne peut pas gérer une terminaison OLT supplémentaire.
TooManyRecords	Le nombre d'articles sélectionné pour extraction produit une réponse à la demande qui dépasse une longueur prédéterminée.
UncontrolledTestInProgress	L'essai d'autovérification ne peut pas être annulé en raison d'un essai non commandé.
UnknownBackupProcess	L'objet nommé de suivi du transfert, identifiant le processus de transfert de fichiers, est inconnu du système de gestion par fournisseur.
UnknownConnection	La connexion de sous-réseau est inconnue du système de gestion du fournisseur.
UnknownDestinationServer	Le serveur de destination désigné ne peut pas être consulté par l'agent de transfert.
UnknownHistoryDataType	Le type de données chronologiques est inconnu dans le système de gestion par fournisseur.
UnknownManagedEntity	L'entité gérée spécifiée est inconnue du système de gestion par fournisseur.
UnknownMonitoringPointTypes	Le type de point de surveillance est inconnu du système de gestion par fournisseur.
UnknownNE	L'élément de réseau identifié est inconnu du système de gestion par fournisseur.
UnknownOption	La valeur indiquée de l'état administratif de l'archive est "ShuttingDown".
UnknownParameters	Le paramètre surveillé qui a été indiqué est inconnu dans le système de gestion par fournisseur.
UnknownPort	L'accès désigné est inconnu du système de gestion par fournisseur.
UnknownProfiles	Cette exception est déclenchée si le nom de profil fourni est inconnu du système de gestion par fournisseur et ne peut pas être extrait du répertoire d'objets de profil.
UnknownRecordSet	La chaîne d'articles désignée dans la demande est inconnue du système de gestion par fournisseur.
UnknownReservationId	Le système de gestion par fournisseur ne reconnaît pas cet identificateur de réservation.
UnknownRestoreProcess	L'objet nommé de suivi du transfert identifiant le processus de transfert de fichiers est inconnu du système de gestion par fournisseur.

Tableau B.1/Q.834.4 – Exceptions

Exception déclenchées	Description
UnknownScheduler	Le programmateur nommé est inconnu du système de gestion par fournisseur.
UnknownService	Le service décrit est inconnu du système de gestion par fournisseur.
UnknownServiceInstance	L'instance de service est inconnue du système de gestion par fournisseur.
UnknownSlot	Le logement requis est inconnu dans l'élément de réseau.
UnknownSoftwareDownloadTrackObject	La charge logicielle nommée (concernant le logiciel distribué à un élément de réseau) est inconnue du système de gestion par fournisseur.
UnknownSoftwareLoad	La charge logicielle spécifiée ne peut pas être trouvée.
UnknownSourceServer	Le système de gestion par fournisseur et/ou OLT ne peut pas communiquer avec le serveur de départ. L'adresse du réseau de communication de données (RCD) est inconnue ou l'accès est bloqué.
UnknownSystemTimingSource	La référence temporelle externe est inconnue du système de gestion par fournisseur.
UnknownTargets	Liste d'activités cibles inconnues du système de gestion par fournisseur.
UnknownTest	Le test spécifié par l'ID d'objet de suivi de test est inconnu du système de gestion par fournisseur.
UnknownTransferProcess	L'état d'avancement du processus de transfert indiqué n'a pas pu être vérifié parce qu'il est inconnu du système de gestion par fournisseur.
UnknownUserGroupId	Le groupe d'utilisateurs est inconnu du système de gestion par fournisseur.
UnknownUserIds	Cette exception est déclenchée quand aucun identificateur d'utilisateur n'est reconnu.
UnrecognisedVersion	La version d'équipement indiquée ne concorde pas avec des valeurs connues.
UserGroupNotEmpty	Un groupe d'utilisateurs non vide ne peut pas être supprimé.
UserLoginPolicyViolation	L'attribution spécifiée d'un nouveau mot de passe à un utilisateur viole la politique relative à l'identifiant d'utilisateur actuellement mise en œuvre par le système de gestion du fournisseur.

Annexe C

Fichiers en langage IDL

Si les fichiers suivants sont sauvegardés sous forme de fichiers de texte, alors tout sous-ensemble de ces fichiers pourra être compilé avec succès au moyen de tout compilateur IDL-OMG conforme à la spécification 2.1 (ou ultérieure) de l'architecture CORBA à condition que le sous-ensemble comprenne le fichier Q834Common.idl et les fichiers normalisés CosNaming.idl, CosNotifyFilter.idl, CosNotification.idl, X780.idl et CosNotifyComm.idl.[11]

C.1 Q834AccessControl.idl

```
#ifndef __Q834_4_ACCESSCONTROL_DEFINED
#define __Q834_4_ACCESSCONTROL_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module AccessControl {
// Début des définitions issues d'autres fichiers en langage idl

// de Q834Common

     typedef Q834Common::ManagedEntityIdType    ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::AdministrationDomainSeqType
AdministrationDomainSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;

#define AccessDenied Q834Common::AccessDenied

// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales

     struct UserLoginPolicyType {
          short minUserId; // Longueur minimale de l'identificateur
                           d'utilisateur
          short minPassword; // Longueur minimale du mot de passe
          short passwordReuse;
          short loginAttempts;
          long passwordValidity;
          boolean alphanumeric;//?le mot de passe doit-il contenir un mélange de
                                caractères alphanumériques
          boolean specialCharacters;//?le mot de passe doit-il contenir des
                                     caractères spéciaux
          boolean repeatingCharacters; //?le mot de passe doit-il contenir
                                       des caractères répétitifs
          boolean disallowUserId; //interdiction du nom d'utilisateur dans le
                                  mot de passe
     };
```

[11] Voir les références [18] et [19].

```
struct SessionPolicyType {
     short sessionInactiveTime;
     short inactiveUserIdDisableTime;
     short multipleActiveLogins;
};

struct PasswordPolicyType {
     UserLoginPolicyType userLoginPolicy;
     SessionPolicyType sessionPolicy;
};


typedef sequence<UserIdType> UserIdSeqType;

enum ActivityLevelType {
     monitorOnly,  // lecture
     allowedToExecute,  // écriture
     noAccess
};

typedef short ActivityType;



struct TargetActivityType {
     ActivityType type;
     ActivityLevelType activityLevel;
     AdministrationDomainSeqType AdministrationDomainSeq;
};

typedef sequence<TargetActivityType> TargetActivitySeqType;

enum UserLoginPolicyViolationReasonType {
     minUserId,
     minPassword,
     passwordReuse,
     loginAttempts,
     passwordValidity,
     alphanumeric,
     specialCharacters,
     repeatingCharacters,
     disallowUserId
};



     typedef sequence<UserLoginPolicyViolationReasonType>
UserLoginPolicyViolationReasonSeqType;
          typedef sequence<UserLabelType> UserGroupIdSeqType;

     struct UserType {
          UserIdType userId;
          UserGroupIdSeqType userGroupIdSeq;
          TargetActivitySeqType TargetActivitySeq;
     };

     struct UserGroupType {
          UserLabelType userGroupId;
          UserIdSeqType userIdSeq;
          TargetActivitySeqType TargetActivitySeq;
     };

     typedef sequence<UserType> UserSeqType;
     typedef sequence<UserGroupType> UserGroupSeqType;
```

```
// Exceptions locales

exception UnknownUserIds {
     UserIdSeqType userIdSeq;
};
exception DuplicateUserId {};
exception UnknownUserGroupId {};
exception DuplicateUserGroupId {};
exception UnknownTargets {
     TargetActivitySeqType unknownTargetActivities;
};
exception UserGroupNotEmpty {};
exception UserLoginPolicyViolation {
     UserLoginPolicyType userLoginPolicy;
     UserLoginPolicyViolationReasonSeqType reason;
};
// Fin des définitions locales

valuetype AccessControlMgrValueType: itut_x780::ManagedObjectValueType {

     public PasswordPolicyType passwordPolicy; // GET
     public UserSeqType userList; // GET
     public UserGroupSeqType userGroupList; // GET
};


interface AccessControlMgr : itut_x780::ManagedObject {

// définissent les activités
     const short ALL_ACTIVITIES  = 0;
     const short ACCESS_CONTROL_MANAGEMENT = 1;
     const short ALARM_EVENT_CONFIGURATION_MANAGEMENT = 2;
     const short SCHEDULE_ACTIVITY    = 3;
     const short SOFTWARE_DOWNLOAD    = 4;
     const short TEST_CONTROL    = 5;
     const short SYNCHRONIZE_CURRENT_EVENT_LIST   = 6;
     const short SYNCHRONIZE_NE  = 7;
     const short RANGE_NE    = 8;
     const short REGISTER_SYSTEM = 9;
     const short RESERVE_RESOURCES    = 10;
     const short PROFILE_MANAGEMENT   = 11;
     const short PROVISION_NE    = 12;
     const short PROVISION_TELEPHONY_SERVICE     = 13;
     const short PROVISION_PACKETISED_DATA_SERVICES    = 14;
     const short PROVISION_VIDEO_SERVICE   = 15;
     const short PROVISION_LEASED_LINE_SERVICE    = 16;
     const short BULK_TRANSFER   = 17;
     const short HISTORY_DATA_COLLECTION   = 18;
     const short CONTROL_ARCHIVING    = 19;
     const short CONTROL_PERFORMANCE_MONITORING   = 20;
     const short CONFIGURATION_BACKUP_RESTORE    = 21;


// Voir § 9.1.1.1 pour la description du comportement de cette opération

     void setPasswordPolicy(
          in PasswordPolicyType passwordPolicy )
          raises ( AccessDenied);

// Voir § 9.1.1.2 pour la description du comportement de cette opération

     PasswordPolicyType passwordPolicyGet()
```

```
          raises (AccessDenied);

// Voir § 9.1.1.3 pour la description du comportement de cette opération

     UserSeqType userListGet ()
          raises (AccessDenied);

// Voir § 9.1.1.4 pour la description du comportement de cette opération

     UserGroupSeqType userGroupListGet ()
          raises (AccessDenied);

// Voir § 9.1.1.5 pour la description du comportement de cette opération

     UserType userGet (
          in UserIdType userId )
          raises (AccessDenied,
               UnknownUserIds);

// Voir § 9.1.1.6 pour la description du comportement de cette opération

     UserGroupType userGroupGet (
          in UserLabelType userGroupId)
          raises (AccessDenied,
               UnknownUserGroupId);


// Voir § 9.1.1.7 pour la description du comportement de cette opération

     void createUserGroup (
          in UserLabelType userGroupId,
          in TargetActivitySeqType targetAdditions)
          raises (DuplicateUserGroupId,
               UnknownTargets,
               AccessDenied);


// Voir § 9.1.1.8 pour la description du comportement de cette opération

     TargetActivitySeqType modifyUserGroup (
          in UserLabelType userGroupId,
          in TargetActivitySeqType targetAdditions,
          in TargetActivitySeqType targetDeletions)
          raises (UnknownUserGroupId,
               UnknownTargets,
               AccessDenied );


// Voir § 9.1.1.9 pour la description du comportement de cette opération

     void deleteUserGroup (
          in UserLabelType userGroupId)
          raises (AccessDenied,
               UserGroupNotEmpty,
               UnknownUserGroupId );



// Voir § 9.1.1.10 pour la description du comportement de cette opération

     void addUsersToGroup (
          in UserLabelType userGroupId,
          in UserIdSeqType userIdList )
          raises (AccessDenied,
```

```
                UnknownUserGroupId); // les copies des utilisateurs sont
                                     ignorées


// Voir § 9.1.1.11 pour la description du comportement de cette opération

     void deleteUsersFromGroup (
          in UserLabelType userGroupId,
          in UserIdSeqType userIdList )
          raises (AccessDenied,
          UnknownUserGroupId,
          UnknownUserIds);


// Voir § 9.1.1.12 pour la description du comportement de cette opération

     TargetActivitySeqType getPermissionList (
          in UserIdType userId )
          raises (UnknownUserIds,
          AccessDenied) ;


// Voir § 9.1.1.13 pour la description du comportement de cette opération

     TargetActivitySeqType modifyPermissionList (
          in UserIdType userId,
          in TargetActivitySeqType targetAdditions,
          in TargetActivitySeqType targetDeletions )
          raises (UnknownUserIds,
               UnknownTargets,
               AccessDenied);

// Voir § 9.1.1.14 pour la description du comportement de cette opération

     void createUser (
          in UserIdType userId,
          in PasswordType password,
          in TargetActivitySeqType targetAdditions )
          raises (DuplicateUserId,
               UnknownTargets,
               AccessDenied,
               UserLoginPolicyViolation);



// Voir § 9.1.1.15 pour la description du comportement de cette opération

     void deleteUser (
          in UserIdType userId )
          raises (UnknownUserIds,
               AccessDenied);


// Voir § 9.1.1.16 pour la description du comportement de cette opération

     void resetPassword (
          in UserIdType userId,
          in PasswordType newPassword )
          raises (UnknownUserIds,
          UserLoginPolicyViolation,
          AccessDenied);

}; // interface AccessControlMgr
```

```
}; // module AccessControl

}; // module q834_4

#endif
```

C.2 Q834Build.idl

```
#ifndef __Q834_4_BUILD_DEFINED
#define __Q834_4_BUILD_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module Build {

// début des définitions issues d'autres fichiers en langage idl

// de Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;

     typedef Q834Common::SlotAssignmentSeqType SlotAssignmentSeqType;
     typedef Q834Common::SerialNumType SerialNumType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ExternalTimeType ExternalTimeType;
     typedef Q834Common::SystemTimingType SystemTimingType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::ReservationIdSeqType ReservationIdSeqType;
     typedef Q834Common::AdministrationDomainType AdministrationDomainType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::LoopbackLocationIdSeqType LoopbackLocationIdSeqType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::ServiceCategoryType ServiceCategoryType;
     typedef Q834Common::ConformanceDefType ConformanceDefType;
     typedef Q834Common::ATMOverbookingFactorType ATMOverbookingFactorType;


#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define InvalidSerialNumSyntax Q834Common::InvalidSerialNumSyntax
#define UnknownProfiles Q834Common::UnknownProfiles
#define InvalidUserLabelSyntax Q834Common::InvalidUserLabelSyntax
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define ParameterViolation Q834Common::ParameterViolation
#define ProfileSuspended Q834Common::ProfileSuspended
#define UnknownNE Q834Common::UnknownNE

// Fin des définitions issues d'autres fichiers en langage idl

     // Types de données locales



     enum NEKindType {
          BPONOLT,
          BPONONT,
          BPONONU,
```

```
        BPONNT
    };



    struct ProtectionUnit
    {
         ManagedEntityIdType portId;
         boolean protectingInd; //TRUE = Protégeant, FALSE = Protégé
    };

    typedef sequence<ProtectionUnit> ProtectionUnitSeqType;

    enum ProtectionRatioType
    {
        oneForOne,
        oneForN,       // N > 1
        mForN          // M > 1
    };
              enum AllowedProtectionSwitchingType
    {
        automatic,
        manual,
        both
    };
    };

       struct ProtectionParameterType
       {
         ProtectionRatioType          protectionRatio;
         AllowedProtectionSwitchingType         protectionSwitchingMethod;
         boolean             revertiveInd;       // TRUE = inversion automatique
         long                waitToRestore;      // nombre de millisecondes
                                                 // avant inversion automatique
       };

    enum SegmentEndpointIndType
    {
        segment,
        endpoint,
        none
    };

enum DirectionType
(
              unidirection,
              bidirection
};



struct OpticalWaveLengthArrayType {
              unsigned long wavelength;
              DirectionType direction;
    };
    typedef sequence <OpticalWaveLengthArrayType>
OpticalWaveLengthArraySeqType;


    // Exceptions locales

    exception InvalidExternalTime {};
    exception UnrecognisedVersion {};
```

```
     exception DuplicateSerialNumber {};
     exception InvalidSlotAssignmentList {};
     exception RemainingContainedManagedEntities {
               ManagedEntityIdSeqType containedManagedEntities ;
     };
     exception UnknownNE {};
     exception UnknownSystemTimingSource {};
     exception RemainingReservations {
          ReservationIdSeqType remainingReservationList;
     };
     exception InvalidEquipmentCode {};
     exception SlotAlreadyAssigned {};
     exception UnknownSlot {};
     exception RemainingSubnetworkConnections {
          ManagedEntityIdSeqType containedManagedEntities ;
     };
          exception InterfaceSpeedNotChangeable {};
     exception InvalidProtectionScheme {};


     // Fin des définitions locales

     valuetype BuilderValueType: itut_x780::ManagedObjectValueType {

          public ManagedEntityIdSeqType createdNodes; // GET
     };

     interface Builder: itut_x780::ManagedObject {


     // Voir § 9.2.1.1 pour la description du comportement de cette opération


ManagedEntityIdType buildNode (
               in NEKindType nEKind,
               in string supplierName,
               in string location,
               in VersionType hWVersion,
               in SerialNumType serialNum,
               in NameSeqType alarmSeverityProfiles,
               in NameSeqType thresholdDataProfiles,
               in SlotAssignmentSeqType slotAssignmentList,
               in ManagedEntityIdType port, // OLT PON port
               in string modelCode,
               in string systemTitle,
               in VersionSeqType softwareVersions,
               in UserLabelType nEUserLabel,
               in ExternalTimeType externalTime,
               in SystemTimingType systemTiming,
               in AdministrationDomainType administrationDomain )
               raises (UnrecognisedVersion,
                    InvalidSerialNumSyntax,
                    DuplicateSerialNumber,
                    UnknownProfiles,
                    UnknownManagedEntity,
                    DuplicateUserLabel,
                    AccessDenied,
                    InvalidExternalTime,
                    UnknownSystemTimingSource,
                    ProfileSuspended);

     // Voir § 9.2.1.2 pour la description du comportement de cette opération

          void assignUserLabelsToNE (
```

```
                in SerialNumType serialNum,
                in UserLabelType nEUserLabel,
                in AdministrationDomainType administrationDomain)
                raises (InvalidSerialNumSyntax,
                     DuplicateSerialNumber,
                     DuplicateUserLabel,
                     AccessDenied);

     // Voir § 9.2.1.3 pour la description du comportement de cette opération

void modifyNode (
                in ManagedEntityIdType managedEntityId,
                in SlotAssignmentSeqType newSlotAssignmentList,
                in NameSeqType newAlarmSeverityProfiles,
                in NameSeqType newThresholdDataProfiles,
                in ManagedEntityIdType port, // OLT PON Port
                in string newModelCode,
                in UserLabelType newNEuserLabel,
                in ExternalTimeType externalTime,
                in AdministrationDomainType administrationDomain )
                raises (UnknownManagedEntity,
                     UnknownNE,
                     InvalidSlotAssignmentList,
                     UnknownProfiles,
                     DuplicateUserLabel,
                     AccessDenied,
                     InvalidExternalTime,
                     ProfileSuspended);

     // Voir § 9.2.1.4 pour la description du comportement de cette opération

          void deleteNode (
                in ManagedEntityIdType managedEntityId )
                raises (UnknownNE,
                     RemainingContainedManagedEntities,
                     AccessDenied,
                     RemainingReservations,
                     RemainingSubnetworkConnections);



     // Voir § 9.2.1.5 pour la description du comportement de cette opération

          void modifyPort (
                in ManagedEntityIdType physicalPathTPId,
                in NameSeqType newAlarmSeverityProfiles,
                in NameSeqType newThresholdDataProfiles,
                in NameSeqType newPortProfiles,
                in string newFrameFormat,
                in AdministrativeStateType administrativeState,
                in OpticalWaveLengthArraySeqType newOpticalWavelengthArray,
                in LoopbackLocationIdSeqType newLoopbackLocationId,
                in unsigned long newInterfaceSpeed,
                in unsigned long aRCTimer)
                raises (UnknownManagedEntity,
                     UnknownProfiles,
                     AccessDenied,
                     InterfaceSpeedNotChangeable,
                     ProfileSuspended);

     // Voir § 9.2.1.6 pour la description du comportement de cette opération

          ManagedEntityIdType buildPlugInUnit (
                in ManagedEntityIdType nEId,
```

```
          in NameType alarmSeverityProfile,
          in UserLabelType plugInUnitUserLabel,
          in string modelCode,
          in ManagedEntityIdType equipmentHolder,
          in AdministrativeStateType administrativeState)
          raises (UnknownNE,
               DuplicateUserLabel,
               AccessDenied,
               UnknownManagedEntity,
               InvalidEquipmentCode,
               SlotAlreadyAssigned,
               UnknownSlot,
               InvalidSlotAssignmentList,
               UnknownProfiles,
               ProfileSuspended);

// Voir § 9.2.1.7 pour la description du comportement de cette opération

     ManagedEntityIdType modifyPlugInUnit (
          in ManagedEntityIdType plugInUnitId,
          in NameType newAlarmSeverityProfile,
          in string newModelCode,
          in ManagedEntityIdType newEquipmentHolder,
          in UserLabelType newPlugInUnitUserLabel,
          in AdministrativeStateType newAdministrativeState)
          raises (UnknownManagedEntity,
               UnknownProfiles,
               AccessDenied,
               InvalidEquipmentCode,
               SlotAlreadyAssigned,
               UnknownSlot,
               InvalidSlotAssignmentList,
               InvalidUserLabelSyntax,
               ProfileSuspended);



// Voir § 9.2.1.8 pour la description du comportement de cette opération

     void deletePlugInUnit (
          in ManagedEntityIdType plugInUnitId )
          raises (UnknownManagedEntity,
               RemainingSubnetworkConnections,
               AccessDenied,
               RemainingReservations);

// Voir § 9.2.1.9 pour la description du comportement de cette opération

     ManagedEntityIdType buildProtectionGrouping(
          in ProtectionParameterType protectionParameters,
          in ProtectionUnitSeqType protectionUnitList)
          raises (InvalidProtectionScheme,
               AccessDenied);

// Voir § 9.2.1.10 pour la description du comportement de cette opération

     void modifyProtectionParameters(
          in ManagedEntityIdType protectionGroupingId,
          in ProtectionParameterType newProtectionParameters)
          raises (UnknownManagedEntity,
               InvalidProtectionScheme,
                  AccessDenied);

// Voir § 9.2.1.11 pour la description du comportement de cette opération
```

```
     void modifyProtectionUnitList(
          in ManagedEntityIdType protectionGroupingId,
          in ProtectionUnitSeqType deltaProtectionUnitList,
          in boolean addDeleteInd)          // Vrai = adjonction
          raises (UnknownManagedEntity,
               InvalidProtectionScheme,
                  AccessDenied);

// Voir § 9.2.1.12 pour la description du comportement de cette opération
     void deleteProtectionGrouping(
          in ManagedEntityIdType protectionGroupingId)
          raises (UnknownManagedEntity,
                  AccessDenied);

// Voir § 9.2.1.13 pour la description du comportement de cette opération
     ManagedEntityIdType buildBridge(
          in NameType mACBridgeProfile,
          in ManagedEntityIdType uplinkPort,
          in ManagedEntityIdSeqType uNIPortList)
          raises (UnknownProfiles,
               AccessDenied,
               UnknownManagedEntity,
               ProfileSuspended);

// Voir § 9.2.1.14 pour la description du comportement de cette opération
     void modifyBridgeProfile(
          in ManagedEntityIdType bridgeId,
          in NameType newMACBridgeProfile)
          raises (UnknownProfiles,
               AccessDenied,
               UnknownManagedEntity,
               ProfileSuspended);

// Voir § 9.2.1.15 pour la description du comportement de cette opération
     void modifyBridgePortList(
          in ManagedEntityIdType bridgeId,
          in ManagedEntityIdSeqType deltaUNIPortList,
          in boolean addDeleteInd)
          raises (AccessDenied,
               RemainingSubnetworkConnections,
               UnknownManagedEntity);

// Voir § 9.2.1.16 pour la description du comportement de cette opération
     void deleteBridge(
          in ManagedEntityIdType bridgeId)
          raises (AccessDenied,
               RemainingSubnetworkConnections,
               UnknownManagedEntity);

// Voir § 9.2.1.17 pour la description du comportement de cette opération
     ManagedEntityIdType buildVPNetworkCTP(
          in ManagedEntityIdType port,
          in short vPI,
          in NameType trafficDescriptorProfileName,
          in ATMOverbookingFactorType overbookingFactor,
          in UserLabelType userLabel,
          in SegmentEndpointIndType segmentEndpointInd)
          raises (AccessDenied,
               UnknownManagedEntity,
               UnknownProfiles,
               ParameterViolation,
               ProfileSuspended);
```

```
     // Voir § 9.2.1.18 pour la description du comportement de cette opération
          void deleteVPNetworkCTP (
               in ManagedEntityIdType vPNetworkCTP)
               raises (AccessDenied,
                    RemainingSubnetworkConnections,
                    UnknownManagedEntity);

     // Voir § 9.2.1.19 pour la description du comportement de cette opération

          ManagedEntityIdSeqType createdNodesGet ()
               raises (AccessDenied);


     }; // interface Builder

}; // module Construction

}; // module q834_4

#endif
```

C.3 Q834Common.idl

```
#ifndef __Q834_4_COMMON_DEFINED
#define __Q834_4_COMMON_DEFINED

#include "CosNaming.idl"
#include "CosNotifyFilter.idl"
#include "itut_x780.idl"
#include "TimeBase.idl"


#pragma prefix "itu.Int"

module q834_4 {

module Q834Common {

// Début des définitions issues d'autres fichiers en langage idl

// de CosNaming
     typedef CosNaming::Name NameType;

// de CosNotifyFilter
     typedef CosNotifyFilter::Filter FilterType;

// de TimeBase
     typedef TimeBase::UtcT UtcT;

// de X780

     typedef itut_x780::ProbableCauseType ProbableCauseType;
     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::OperationalStateType OperationalStateType;
     typedef itut_x780::ProceduralStatusSetType ProceduralStatusSetType;
     typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
     typedef itut_x780::AvailabilityStatusSetType AvailabilityStatusSetType;
     typedef itut_x780::UsageStateType UsageStateType;
     typedef itut_x780::ControlStatusSetType ControlStatusSetType;
     typedef itut_x780::SpecificProblemSetType SpecificProblemSetType;
     typedef itut_x780::BackedUpStatusType BackedUpStatusType;
     typedef itut_x780::AttributeChangeSetType AttributeChangeSetType;
     typedef itut_x780::SecurityAlarmCauseType SecurityAlarmCauseType;
```

```
// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales

     struct NamingComponentType {
          string type;  // type d'entité gérée
          string id;
     };

     typedef sequence<NamingComponentType> RDNType;
     typedef sequence<RDNType> RDNSeqType;
     typedef sequence<NameType>  NameSeqType;

     enum IdType {
          none,
          x780_fineGrained,
          x780_coarseGrained
     };

     typedef RDNType MEIdType;

     struct ManagedEntityIdType {
          IdType id;
          MEIdType mEId;
     };

     typedef sequence<ManagedEntityIdType> ManagedEntityIdSeqType;
     typedef string UserLabelType;
     typedef string SerialNumType;
     typedef string VersionType;
     typedef string PlugInUnitType;
     typedef sequence<UserLabelType> UserLabelSeqType;

     typedef UserLabelType AdministrationDomainType;
     typedef sequence<AdministrationDomainType> AdministrationDomainSeqType;

     typedef string DCNAddressType;

     typedef unsigned short RecordKindType;


     typedef   string    PlugInType;

     // SlotAssignmentList
     struct SlotAssignmentType {
          short number;
          PlugInType plugIn; // une chaîne vide implique que le
                             // mini-logement n'est pas préfourni pour
                             // un type particulier de module
                             // d'extension
     };

     typedef sequence <SlotAssignmentType> SlotAssignmentSeqType;
     typedef string ReservationIdType;
     typedef sequence<ReservationIdType> ReservationIdSeqType;
     typedef string ServiceInstanceIdType;
     typedef sequence<ServiceInstanceIdType> ServiceInstanceIdSeqType;

     typedef UtcT GeneralizedTimeType;
     typedef GeneralizedTimeType ExternalTimeType;

     enum SystemTimingSourceType {
          internalTimingSource,   // autonome
```

```
        remoteTimingSource,     // externe
        slaveTimingTerminationSignal
    };

    typedef ManagedEntityIdType TimingSourceIdType ;

    struct SystemTimingComponentType {
        SystemTimingSourceType type;
        TimingSourceIdType  id;
    };

    struct SystemTimingType {
        SystemTimingComponentType    Primary;
        SystemTimingComponentType    Secondary;
    };


    typedef UserLabelType UserIdType;
    typedef string PasswordType;

    /*
Une série de zéros indique une demande de rebouclage orientée vers tous les
points de connexion ayant un identificateur LLID dans le flux. Une série
d'unités indique une demande de rebouclage orientée vers l'extrémité (extrémité
de segment ou de chaîne de connexion).
'x6A'H indique l'absence d'extrémité CP désignée pour le rebouclage, et donc
aucun rebouclage ne devrait être effectué. Toutes les autres valeurs pour LLID
indiquent une demande de rebouclage orientée vers un emplacement
d'identificateur LLID spécifique.
    */
    typedef   sequence<octet,16> LoopbackLocationIdSeqType;

    typedef string FilenameType; // spécifie le chemin complet

    enum StatusType {
        procedural_status,
        other
    };

    enum OtherStatusType {
        completed_success,
        completed_failure,
        activityInProgress,
        unknownstatus
    };

    union StatusValueType switch (StatusType) {
        case procedural_status: ProceduralStatusSetType proceduralStatusSet;
        case other: OtherStatusType otherStatus;
    };

    struct StatusAttributeType {
        StatusValueType valueOfStatus;
        ManagedEntityIdType ne;
        short percentComplete;
    };

    typedef sequence<StatusAttributeType>  StatusAttributeSeqType;

    typedef   unsigned long  TrackingObjectIdType;

    typedef TrackingObjectIdType TransferTrackingObjectIdType;
```

```
    typedef any RecordType; // définition fondée sur le type d'article
    journalisé (Valeurs définies dans Q834Common::RecordSetType). Nécessite la
    définition du type d'article et du type individuel des articles
    typedef sequence<RecordType> RecordSeqType;


    typedef unsigned long long NotificationIdentifierType; // cette
                      // définition sera remplacée par NotifIDType
                      // définie dans la Rec. UIT-T X.780 quand cette
                      // définition sera changée de "long" à "long
                      // long"

    typedef sequence<NotificationIdentifierType>
NotificationIdentifierSeqType;


typedef stringMonitoredParameterType; // Valeurs définies dans
Q834Common::MonitoringParameter
    typedef unsigned short MonitoringKindType; // valeurs définies dans
Q834Common::PMCategory interface

    struct EndPointType {
        ManagedEntityIdType portId;
        any endPointParameters;
        NameSeqType serviceCharacteristicsProfiles;
    };

    /*
    endPointParameters: paramètres spécifiques de service, structures qui
    doivent être fournies dans le cadre de l'implémentation. Une connexion ATM
    par exemple aurait les paramètres d'identificateur VPI,VCI.
    */


    enum AlarmStatusComponentType  {
        AS_UnderRepair,
        AS_Critical,
        AS_Major,
        AS_Minor,
        AS_AlarmOutstanding
    };

    typedef sequence<AlarmStatusComponentType>  AlarmStatusSeqType;

    struct   EquipmentHolderAddressType {
        short    shelfNumber;
        short    slotNumber;
    };

    /*
    Si le logement d'équipement est une alvéole, le numéro de logement est 0.
    Si le logement d'équipement est un mini-logement, alors les numéros de
    logement comme d'alvéole sont supérieurs ou égaux à 1.
    */

    /*
    Le fournisseur donne une personnalisation de l'interface
    DiscoveryEventSupplier en définissant les noms de module d'extension par
    des chaînes constantes. Il appartiendra à l'opérateur d'assurer la
    cohérence et la non-redondance entre de multiples solutions de
    fournisseur.
    */

    typedef   sequence<PlugInUnitType>    PlugInUnitSeqType;
```

```
struct    NEFSANType {
          ManagedEntityIdType     managedEntityId;
          AdministrativeStateType administrativeState;
          OperationalStateType    operationalState;
          GeneralizedTimeType     externalTime;
          string    locationName;
          string    supplierName;
          VersionType   hardwareVersion;
          VersionSeqType softwareVersions;
          string    serialNumber;
          NameSeqType   alarmSeverityAssignmentProfileNames;
          AlarmStatusSeqType alarmStatusValues;
          NameSeqType   thresholdDataNames;
          ManagedEntityIdSeqType  supportedByManagedEntityList;
          UserLabelType userLabel;
     };

     /*
     La liste SupportedByManagedEntityList devrait comprendre l'instance des
     logiciels contrôlant l'élément de réseau.
     */

     /*
     L'on définit ensuite les structures qui forment la base des informations
     d'inventaire d'équipement découvertes à l'installation ou lors d'actions de
     remaniement de l'équipement. Chacune de ces structures sera ultérieurement
     identifiée comme une structure d'entité gérée dans la spécification de
     l'interface DiscoveryEventSupplier.
     */

     struct    OLTType   {
          NEFSANType    nEFSAN;
          ManagedEntityIdSeqType  subtendingNEFSANList;
     };

     struct    ONTType   {
          NEFSANType    nEFSAN;
          ManagedEntityIdType     upstreamNEFSAN;
     };

     struct    ONUType   {
          NEFSANType    nEFSAN;
          ManagedEntityIdType     upstreamNEFSAN;
          ManagedEntityIdSeqType  subtendingNEFSANList;
     };

     struct    NTType    {
          NEFSANType    nEFSAN;
          ManagedEntityIdType     upstreamNEFSAN;
     };

     struct    EquipmentHolderFType    {
          ManagedEntityIdType     equipmentHolderFId;
          ManagedEntityIdType     containingNEId;
          EquipmentHolderAddressType  equipmentHolderAddress;
          boolean   slotStatus;
          PlugInUnitSeqType  expectedPlugInUnits;
          string    SoftwareLoad;
          NameType  alarmSeverityAssignmentProfileName;
          AlarmStatusSeqType alarmStatusValues;
          OperationalStateType    operationalState;
     };
```

```
struct   PlugInUnitFType    {
     ManagedEntityIdType     plugInUnitFId;
     ManagedEntityIdType     containingNEId;
     EquipmentHolderAddressType  containingSlotAddress;
     AdministrativeStateType administrativeState;
     AvailabilityStatusSetType   availabilityStatus;
     OperationalStateType    operationalState;
     string    modelCode;
     string    functionCode;
     string    supplierName;
     VersionType   hardwareVersion;
     string    serialNumber;
     short          portCount;
     NameSeqType   alarmSeverityAssignmentProfileNames;
     NameSeqType   thresholdDataNames;
     UserLabelType circuitPackUserLabel;
     ManagedEntityIdSeqType  supportedByManagedEntityList;
};

enum  ServiceCategoryType {
     CBR,
     UBR,
     RTVBR,
     NRTVBR,
     AdaptiveBR,
     GFR
};

enum ConformanceDefType {
     CBR1,
     UBR1,
     UBR2,
     VBR1,
     VBR2,
     VBR3,
     ABR,
     GFR1,
     GFR2
};

struct ATMOverbookingFactorType {
     ServiceCategoryType serviceCategory;
     ConformanceDefType  conformanceDef;
     short             overbookingFactor;  // pourcentage: 100 = aucune
                                           // surréservation
};

struct AlarmLogRecordType {
     long long recordId;
     ManagedEntityIdType mEId;
     GeneralizedTimeType loggingTime;
     short probableCause;
     PerceivedSeverityType severity;
     GeneralizedTimeType eventTime;
     ManagedEntityIdType backupEntityId;
     boolean backedupStatus;
     boolean serviceAffectingInd;
     ServiceInstanceIdSeqType affectedServices;
     NotificationIdentifierType notificationId;
     NotificationIdentifierSeqType correlatedNotifications;
     string monitoredParameter;
     long long thresholdValue;
     long long observedValue;
```

```
    string additionalText;
};

struct SecurityAlarmLogRecordType {
    long long recordId;
    ManagedEntityIdType mEId;
    GeneralizedTimeType loggingTime;
    short securityAlarmCause;    // valeurs définies dans la
                                 // Rec. UIT-T X.780
    GeneralizedTimeType eventTime;
    ManagedEntityIdType securityAlarmDetector;
    ManagedEntityIdType serviceUser;
    ManagedEntityIdType serviceProvider;
    NotificationIdentifierType notificationId;
    NotificationIdentifierSeqType correlatedNotifications;
    string additionalText;
};

struct ServiceOutageRecordType {
    long long recordId;
    ServiceInstanceIdType affectedService;
    GeneralizedTimeType loggingTime;
    GeneralizedTimeType outageStartTime;
    GeneralizedTimeType outageEndTime;
    NotificationIdentifierSeqType correlatedNotifications;
};

struct AAL1PMHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long HeaderErrors;
    unsigned long long LostCells;
    unsigned long long CellMisinsertion;
    unsigned long long BufferUnderflows;
    unsigned long long SequenceViolations;
    unsigned long long SDTPtrReframes;
    unsigned long long SDTPtrParityCheckFailures;
};

struct AAL2PMHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;        unsigned long long CPSInPkts;
    unsigned long long CPSOutPkts;
    unsigned long long BufferUnderflow;
    unsigned long long BufferOverflow;
    unsigned long long ParityErrors;
    unsigned long long SeqNumErrors;
    unsigned long long CPS_OSFMismatchErrors;
    unsigned long long CPS_OSFErrors;
    unsigned long long CPSHECErrors;
    unsigned long long OversizedSDUErrors;
    unsigned long long ReassemblyErrors;
    unsigned long long HECOverlapErrors;
    unsigned long long UUIErrors;
    unsigned long long CIDErrors;
};

struct AAL5PMHistoryDataType {
```

```
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long SumOfInvalidCSFieldErrors;
        unsigned long long CRCViolations;
        unsigned long long BufferOverflows;
        unsigned long long EncapProtocolErrors;
    };

    struct APONPMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long ES;
        unsigned long long FEES;
    };

    struct ATMTrafficLoadHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long CellsReceived;
        unsigned long long CellsTransmitted;
    };

    struct DS1PMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long ESP;
        unsigned long long BESP;
        unsigned long long SESP;
        unsigned long long UASP;
        unsigned long long ESPFE;
        unsigned long long BESPFE;
        unsigned long long SESPFE;
        unsigned long long UASPFE;
    };

    struct DS3PMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long ESL;
        unsigned long long SESL;
        unsigned long long CVCPorCVPP;
        unsigned long long ESCPPorESPP;
        unsigned long long SESCPPorSESPP;
        unsigned long long UASCPPorUASPP;
    };

    struct E1PMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
```

```
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ESP;
     unsigned long long BESP;
     unsigned long long SESP;
     unsigned long long UASP;
     unsigned long long ESPFE;
     unsigned long long BESPFE;
     unsigned long long SESPFE;
     unsigned long long UASPFE;
};

struct EthernetHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long SingleCollisionFrame;
     unsigned long long MultipleCollisionFrames;
     unsigned long long SQE;
     unsigned long long DeferredTransmission;
     unsigned long long LateCollision;
     unsigned long long ExcessiveCollision;
     unsigned long long InternalMACTransmitError;
     unsigned long long CarrierSenseError;
     unsigned long long BufferOverflows;
     unsigned long long AlignmentError;
     unsigned long long FrameTooLongs;
     unsigned long long FCSErrors;
     unsigned long long InternalMACReceiveError;
};

struct MACBridgePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long NumberofSuppressedIntervals;
     unsigned long long BridgeLearningEntryDiscard;
};

struct MACBridgePMPortHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ForwardedFrame;
     unsigned long long DelayExceededDiscard;
     unsigned long long MTUExceededDiscard;
     unsigned long long ReceivedFrame;
     unsigned long long ReceivedAndDiscarded;
};

struct UpcNpcDisagreementPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long DiscardedCells;
```

```
     unsigned long long DiscardedCLP_0Cells;
     unsigned long long TaggedCLP_0Cells;
};

struct VoicePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long IncomingCallAttempts;
     unsigned long long OutgoingCallAttempts;
     unsigned long long VoicePortBufferOverflows;
     unsigned long long VoicePortBufferUnderflows;
};

struct VpVcPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long Lost0plus1UserInformationCells;
     unsigned long long Lost0UserInformationCells;
     unsigned long long MisinsertedUserInformationCells;
     unsigned long long Transmitted0plus1UserInformationCells;
     unsigned long long Transmitted0UserInformation;
     unsigned long long ImpairedBlock;
};

struct SONETSDHLinePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ErroredSecondsP;
     unsigned long long SeverelyErroredSecondsP;
     unsigned long long BackgroundBlockErrorP;
     unsigned long long OutOfFrameSecondsP;
     unsigned long long UnavailableSecondsP;
};

struct SONETSDHSectionPathPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ErroredSecondsP;
     unsigned long long SeverelyErroredSecondsP;
     unsigned long long BackgroundBlockErrorP;
     unsigned long long OutOfFrameSecondsP;
     unsigned long long UnavailableSecondsP;
     unsigned long long FailureCountP;
     unsigned long long ErroredSecondsTypeAP;
     unsigned long long ErroredSecondsTypeBP;
     unsigned long long ErroredSecondsPFE;
     unsigned long long SeverelyErroredSecondsPFE;
     unsigned long long BackgroundBlockErrorPFE;
     unsigned long long OutOfFrameSecondsPFE;
     unsigned long long UnavailableSecondsPFE;
     unsigned long long FailureCountPFE;
     unsigned long long ErroredSecondsTypeAPFE;
```

```
        unsigned long long ErroredSecondsTypeBPFE;
    };

    struct SONETSDHSectionAdaptationPMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long PointerJustificationHighCountP;
        unsigned long long PointerJustificationLowCountP;
    };

    struct TCAdaptationProtocolMonitoringPMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long DiscardedCellsHECViolationP;
        unsigned long long ErroredCellsHECViolationP;
    };

// Exceptions communes

    exception AccessDenied {};
    exception CommFailure {};
    exception ConnectionCountExceeded {};
    exception DuplicateUserLabel {};
    exception EquipmentFailure {};
    exception InsufficientBW {};
    exception InsufficientPONBW {
        ManagedEntityIdType ponPORT;
    };
    exception InvalidSerialNumSyntax {};
    exception InvalidUserLabelSyntax {};
    exception MaxSubtendingNodesExceeded {};
    exception Timeout {};
    exception UnknownManagedEntity {
        ManagedEntityIdType  managedEntityId;
    };
    exception UnknownNE {
        ManagedEntityIdSeqType unknownNEs;
    };
    exception UnknownPort {};
    exception UnknownProfiles  {};
    exception UnknownReservationId {};
    exception UnknownScheduler {};
    exception InvalidScheduler {};
    exception DCNTimeout {};
    exception UnknownDestinationServer {};
    exception UnknownServiceInstance {};
    exception DeniedAccess {};
    exception BackupInProgress {};
    exception InvalidStartTime {};
    exception InvalidStopTime {};
    exception ParameterViolation {};
    exception UnknownRecordSet {};
    exception SynchInProgress {};
    exception ProfileSuspended {
        NameSeqType profilesSuspended;
    };
```

```
    module ProbableCauseConst {

        const string moduleName = "q834_4::Q834Common::ProbableCauseConst";

        interface ProbableCause {
        /*
        Pour X.780 probableCause voir itut_x780::ProbableCauseConst.
        */

        // Valeurs additionnelles de cause probable requises pour BPON


            const unsigned short LOSS_OF_PATH_POINTER = 1;
            const unsigned short STS_PAYLOAD_LABEL_MISMATCH = 2;
            const unsigned short STS_PATH_UNEQUIPPED = 3;
            const unsigned short ALARM_INDICATION_SIGNAL = 4;
            const unsigned short REMOTE_FAILURE_INDICATION = 5;
            const unsigned short REMOTE_ALARM_INDICATION = 6;
            const unsigned short ALARM_INDICATION_SIGNAL_PATH = 7;
            const unsigned short
ALARM_INDICATION_SIGNAL_CUSTOMER_INSTALLATION = 8;
            const unsigned short LOSS_OF_SIGNAL_TO_ALL_ONU_ONU = 9;
            const unsigned short LOSS_OF_SIGNAL_ONU_OR_ONT = 10;
            const unsigned short LOSS_OF_ACKNOWLEDGEMENT_ONU_OR_ONT = 11;
            const unsigned short PLOAMCL_ONU_OR_ONT = 12; //Perte de cellules
OAM dans la couche Physique
            const unsigned short LOCD_ONU_OR_ONT = 13; // Perte de cadrage
                                                       // cellulaire
            const unsigned short CPHE_ONU_OR_ONT = 14; // Déphasage cellulaire
            const unsigned short PEE_ONU_OR_ONT = 15;
            const unsigned short RF_ONU_OR_ONT = 16; // Echec télémétrique
            const unsigned short BED_ONU_OR_ONT = 17; // Détection d'erreur de
                                                      // bloc
            const unsigned short SD_ONU_OR_ONT = 18; // Signal dégradé
            const unsigned short REI_ONU_OR_ONT = 19; // Indication d'erreur
                                                      // distante
            const unsigned short UM_ONU_OR_ONT = 20; // Message inconnu
            const unsigned short LM_ONU_OR_ONT = 21; // Liaison discordante
            const unsigned short REMOTE_DEFECT_INDICATION = 22; // Signal
                                                                // dégradé
            const unsigned short MAJOR_POWER_FAILURE = 23;
            const unsigned short
REMOTE_ALARM_INDICATION_FAR_END_CUSTOMER_INSTALLATION = 24;
            const unsigned short LOSS_OF_ATM_CELL_DELINEATION = 25;
            const unsigned short LOW_BATTERY_THRESHOLD = 26;
                 const unsigned short DIAGNOSTIC_TEST_FAILURE = 27; // Voir
                                                                    // commentaire
                                                                    // ci-dessous
            const unsigned short LOSS_OF_DCN_LINK = 28;
            const unsigned short CELL_STARVATION = 29;
            const unsigned short UNEXPECTED_PLUGIN = 30;
            const unsigned short IMPROPER_CARD_REMOVAL = 31;
            const unsigned short SLOT_CARD_MISMATCH = 32;
            const unsigned short LOS_LAN = 33; // Perte de porteuse à un accès
                                               // de pont local
            const unsigned short PERSISTENT_IMPAIRMENT = 34;
        }; // interface ProbableCause
    }; // module ProbableCauseConst

    interface MonitoringParameter {
        /*
        Noms pour les paramètres de contrôle de la qualité du fonctionnement
        et du trafic
        */
```

```
        const    string allParameters = "AllParameters";
        const    string aAL1HeaderErrors = "AAL1HeaderErrors";
        const    string allTypesCellsDiscarded = "AllTypesCellsDiscarded";
        const    string aTMProtocolErrors = "ATMProtocolErrors";
        const    string bESFEP = "BESFEP";
        const    string bESP = "BESP";
        const    string bufferOverflows = "BufferOverflows";
        const    string bufferUnderflows = "BufferUnderflows";
        const    string cellDelineationAnomalies = "CellDelineationAnomalies";
        const    string cellMisinsertion = "CellMisinsertion";
        const    string cRCViolations = "CRCViolations";
        const    string cVCP = "CVCP";
        const    string cVL = "CVL";
        const    string cVPP = "CVPP";
        const    string cVS = "CVS";
        const    string discardedAllTypeCellsduetoNPC =
"DiscardedAllTypeCellsduetoNPC";
        const    string discardedAllTypeCellsduetoUPC =
"DiscardedAllTypeCellsduetoUPC";
        const    string discardedPriorityCellsduetoNPC =
"DiscardedPriorityCellsduetoNPC";
        const    string discardedPriorityCellsduetoUPC =
"DiscardedPriorityCellsduetoUPC";
        const    string encapProtocolErrors = "EncapProtocolErrors";
        const    string eSCPP = "ESCPP";
        const    string eSPONT = "ESPONT";
        const    string eSL = "ESL";
        const    string eSP = "ESP";
        const    string eSPP = "ESPP";
        const    string eSS = "ESS";
        const    string excessiveCollisions = "ExcessiveCollisions";
        const    string fCSErrors = "FCSErrors";
        const    string frameTooLongs = "FrameTooLongs";
        const    string hECViolations = "HECViolations";
        const    string impairedBlocks = "ImpairedBlocks";
        const    string lateCollisions = "LateCollisions";
        const    string lostCells = "LostCells";
        const    string lostPriorityUserInformationCells =
"LostPriorityUserInformationCells";
        const    string lostUserInformationCells = "LostUserInformationCells";
        const    string misinsertedUserInformationCells =
"MisinsertedUserInformationCells";
        const    string priorityCellsDiscarded = "PriorityCellsDiscarded";
        const    string sDTPointerReframes = "SDTPointerReframes";
        const    string sDTPointerParityCheckFailures =
"SDTPointerParityCheckFailures";
        const    string sequenceViolations = "SequenceViolations";
        const    string sESCPP = "SESCPP";
        const    string sESPONT = "SESPONT";
        const    string sESL = "SESL";
        const    string sESP = "SESP";
        const    string sESPP = "SESPP";
        const    string sESS = "SESS";
        const    string sumOfInvalidCSFieldErrors =
"SumOfInvalidCSFieldErrors";
        const    string uASPONT = "UASPONT";
        const    string uASCPP = "UASCPP";
        const    string uASP = "UASP";
        const    string uASPP = "UASPP";
        const    string incomingCallAttempts = "IncomingCallAttempts";
        const    string outgoingCallAttempts = "OutgoingCallAttempts";
        const    string voicePortBufferOverflows = "VoicePortBufferOverflows";
```

```
          const     string voicePortBufferUnderflows =
"VoicePortBufferUnderflows";
          const     string cPSInPkts = "CPSInPkts";
          const     string cPSOutPkts = "CPSOutPkts";
          const     string bufferUnderflow = "BufferUnderflow";
          const     string bufferOverflow = "BufferOverflow";
          const     string parityErrors = "ParityErrors";
          const     string seqNumErrors = "SeqNumErrors";
          const     string cPS_OSFMismatchErrors = "CPS_OSFMismatchErrors";
          const     string cPS_OSFErrors = "CPS_OSFErrors";
          const     string cPSHECErrors = "CPSHECErrors";
          const     string oversizedSDUErrors = "OversizedSDUErrors";
          const     string reassemblyErrors = "ReassemblyErrors";
          const     string hECOverlapErrors = "HECOverlapErrors";
          const     string uUIErrors = "UUIErrors";
          const     string cIDErrors = "CIDErrors";

     }; // interface MonitoringParameter

     interface PMCategory {
          const unsigned short DS1_PM = 1;
          const unsigned short DS3_PM = 2;
          const unsigned short E1_PM = 3;
          const unsigned short E3_PM = 4;
          const unsigned short VP_PM = 5;
          const unsigned short VC_PM = 6;
          const unsigned short ETHERNET_PM = 7;
          const unsigned short TC_ADAPTOR_PM = 8;
          const unsigned short VOICE_PM = 9;
          const unsigned short APON_PM = 10;
          const unsigned short MACBRIDGE_PM = 11;
          const unsigned short MACBRIDGEPORT_PM = 12;
          const unsigned short ATMTRAFFICLOAD_PM = 13;
          const unsigned short AAL1_PM = 14;
          const unsigned short AAL2_PM = 15;
          const unsigned short AAL5_PM = 16;
          const unsigned short UPCNPC_PM = 17;
          const unsigned short DBA_PM = 18;
          const unsigned short SONET_SDH_LINE_PM = 19;
          const unsigned short SONET_SDH_SECTION_PATH_PM = 20;
          const unsigned short SONET_SDH_SECTION_ADAPTATION_PM = 21;
          const unsigned short CALL_STATS_PM = 22;
          const unsigned short VIDEO_PM = 23;

     }; // interface PMCategory

     interface RecordSetType {

     // Début de valeurs pour RecordKindType
     // valeurs 1-99 réservées pour HistoryDataType

          const unsigned short DS1PMHISTORYDATA = 1;
          const unsigned short DS3PMHISTORYDATA = 2;
          const unsigned short E1PMHISTORYDATA = 3;
          const unsigned short E3PMHISTORYDATA = 4;
          const unsigned short VPVCPMHISTORYDATA = 5;
          const unsigned short AAL1PMHISTORYDATA = 6;
          const unsigned short AAL2PMHISTORYDATA = 7;
          const unsigned short AAL5PMHISTORYDATA = 8;
          const unsigned short UPCNPCDISAGREEMENTPMHISTORYDATA = 9;
          const unsigned short ETHERNETPMHISTORYDATA = 10;
          const unsigned short VOICEPMHISTORYDATA = 11;
          const unsigned short MACBRIDGEPORTPMHISTORYDATA = 12;
          const unsigned short MACBRIDGEPMHISTORYDATA = 13;
```

```
          const unsigned short APONPMHISTORYDATA = 14;
          const unsigned short SONETSDHLINEPMHISTORYDATA = 15;
          const unsigned short SONETSDHSECTIONADAPTATIONPMHISTORYDATA = 16;
          const unsigned short SONETSDHSECTIONPATHPMHISTORYDATA = 17;
          const unsigned short TCADAPTATIONPROTOCOLMONITORINGPMHISTORYDATA = 18;
          const unsigned short ALARMLOGRECORD = 100;
          const unsigned short SECURITYALARMLOGRECORD = 101;
          const unsigned short SERVICEOUTAGERECORD = 102;

     // Fin de valeurs pour RecordKindType

     }; // interface RecordSetType

interface   PhysicalLayerLoopback {

     // Début de valeurs pour LoopbackTestType

          const unsigned short LINELOOPBACK = 1;
          const unsigned short PAYLOADLOOPBACK = 2;
          const unsigned short INWARDLOOPBACK = 3;
          const unsigned short DUALLOOPBACK = 4;
          const unsigned short FACILITYLOOPBACK = 5;
          const unsigned short TERMINALLOOPBACK = 6;

     // Fin de valeurs pour LoopbackTestType

     }; // interface  PhysicalLayerLoopback

};   // module Q834Common

}; // module q834_4
#endif
```

C.4 Q834ControlArchive.idl

```
#ifndef __Q834_4_CONTROLARCHIVE_DEFINED
#define __Q834_4_CONTROLARCHIVE_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ControlArchive {

// début des définitions issues d'autres fichiers en langage idl

// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::FilterType FilterType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::OperationalStateType OperationalStateType;
     typedef Q834Common::RecordType RecordType;
     typedef Q834Common::RecordSeqType RecordSeqType;
     typedef Q834Common::UserLabelSeqType UserLabelSeqType;
     typedef Q834Common::RecordKindType RecordKindType;


#define AccessDenied             Q834Common::AccessDenied
#define DuplicateUserLabel       Q834Common::DuplicateUserLabel
```

```
#define Timeout                  Q834Common::Timeout
#define UnknownRecordSet         Q834Common::UnknownRecordSet

// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales

     enum FullActionType {
          halt, // indique que le journal devrait arrêter d'enregistrer de
                // nouveaux articles s'il est plein
          wrap // indique que le journal devrait supprimer d'anciens articles
               // enregistrés s'il est plein.
     };

     typedef unsigned long long MaxSizeType;
     typedef unsigned short SizeThresholdType; // 0-100%
     typedef unsigned long long CurrentSizeType;

     struct RecordSetStatusType {
          CurrentSizeType currentSize;
          OperationalStateType operationalState;
          MaxSizeType maxSize;
          SizeThresholdType sizeThreshold;
          NameType filterName;
          FullActionType fullAction;
          AdministrativeStateType administrativeState;
          RecordKindType recordKind;
          UserLabelType recordSetUserLabel;
     };


     enum CreationModeType {
          operatorDefined,
          initialList,
          either
     };

// Exceptions locales

     exception RecordSetExists {ManagedEntityIdType recordSetId;};
     exception LockedAlready {};
     exception UnknownOption {};
     exception NoSuchRecords {};
     exception TooManyRecords {};

// Fin des définitions locales



     interface RecordSetMgr : itut_x780::ManagedObject {


     // Voir § 9.4.1.1 pour la description du comportement de cette opération

          ManagedEntityIdType createLog (
               in UserLabelType recordSetUserLabel,
               in AdministrativeStateType administrativeState,
               in NameType filterName,
               in FullActionType fullAction,
               in MaxSizeType maxSize,
               in SizeThresholdType sizeThreshold)
               raises (RecordSetExists,
                    DuplicateUserLabel,
                    AccessDenied);
```

```
// Voir § 9.4.1.2 pour la description du comportement de cette opération

     ManagedEntityIdType createArchive (
          in UserLabelType recordSetUserLabel,
          in AdministrativeStateType administrativeState,
          in RecordKindType recordKind,
          in MaxSizeType maxSize)
          raises (RecordSetExists,
               DuplicateUserLabel,
               AccessDenied);

// Voir § 9.4.1.3 pour la description du comportement de cette opération

     RecordSetStatusType getStatusAttributes (
          in ManagedEntityIdType recordSetId)
          raises (AccessDenied, UnknownRecordSet);


// Voir § 9.4.1.4 pour la description du comportement de cette opération

     void suspendArchive (
          in ManagedEntityIdType recordSetId)
          raises (AccessDenied, UnknownRecordSet);

// Voir § 9.4.1.5 pour la description du comportement de cette opération

     void resumeArchive (
          in ManagedEntityIdType recordSetId)
          raises (AccessDenied, UnknownRecordSet);

// Voir § 9.4.1.6 pour la description du comportement de cette opération

     void deleteArchive (
          in ManagedEntityIdType recordSetId )
          raises (UnknownRecordSet,
               AccessDenied );

// Voir § 9.4.1.7 pour la description du comportement de cette opération

     void purgeArchive (
          in ManagedEntityIdType recordSetId )
          raises (UnknownRecordSet,
               AccessDenied );


// Voir § 9.4.1.8 pour la description du comportement de cette opération

     RecordSeqType selectRecords (
          in FilterType SelectionFilter,
          in ManagedEntityIdType recordSetId)
          raises (UnknownRecordSet,
               Timeout,
               NoSuchRecords,
               AccessDenied,
               TooManyRecords);

// Voir § 9.4.1.9 pour la description du comportement de cette opération

     ManagedEntityIdSeqType recordSetListGet (
          in CreationModeType creationMode)
          raises (AccessDenied);
```

```
     // Voir § 9.4.1.10 pour la description du comportement de cette opération

          void changeUserLabel(
               in ManagedEntityIdType recordSetId,
               in UserLabelType newUserLabel)
               raises (UnknownRecordSet,
                    AccessDenied,
                    DuplicateUserLabel);

     }; // interface RecordSetMgr

}; // module ControlArchive

}; // module q834_4

#endif
```

C.5 Q834SoftwareDownload.idl

```
#ifndef __Q834_4_SOFTWAREDOWNLOAD_DEFINED
#define __Q834_4_SOFTWAREDOWNLOAD_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module SoftwareDownload {

// début des définitions issues d'autres fichiers en langage idl

// de Q834Common
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::PlugInUnitType PlugInUnitType;
     typedef Q834Common::FilenameType  FilenameType;
     typedef Q834Common::ProceduralStatusSetType ProceduralStatusSetType;
     typedef Q834Common::StatusAttributeSeqType StatusAttributeSeqType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::TrackingObjectIdType TrackingObjectIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define UnknownNE Q834Common::UnknownNE
#define DeniedAccess Q834Common::DeniedAccess
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define Timeout Q834Common::Timeout
#define InvalidScheduler Q834Common::InvalidScheduler
#define InvalidStartTime Q834Common::InvalidStartTime

// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales


     typedef TrackingObjectIdType SoftwareDownloadTrackingObjectIdType;
```

```
     typedef sequence<SoftwareDownloadTrackingObjectIdType>
SoftwareDownloadTrackingObjectIdSeqType;

     typedef sequence<FilenameType> FilenameSeqType;

     struct VersionsType {
          ManagedEntityIdType     resourceId;
          VersionType softwarePrimary;
          VersionType softwareStandBy;
          VersionType hardware;
     };

     typedef sequence<VersionsType> VersionsSeqType;

     struct TargetType {
          ManagedEntityIdType containingSystem;
          ManagedEntityIdType containingNE; // une séquence vide signifie un
                                            // type quelconque
          string        plugInUnitType; // une chaîne vide implique ici
                                        // tout type approprié de module
                                        // d'extension
          ManagedEntityIdType slot;// une séquence vide implique ici un
                                   // mini-logement quelconque.
     };                        // la chaîne est donnée par le
                               // fournisseur avec les Notes de version.

     struct DownloadStatusType
     {
             ManagedEntityIdType targetId;
             StatusValueType deliveryStatus;
             StatusValueType commitStatus;
             StatusValueType activationStatus;
             StatusValueType revertStatus;
     };

        typedef sequence<DownloadStatusType> DownloadStatusSeqType;

// Exceptions locales


     exception InvalidExternalTime {};
     exception UnrecognisedTarget {};
     exception InsufficientMemory {};
     exception SoftwareLoadHWMismatch {};
     exception SourceUnreachable {};
     exception UnknownSoftwareLoad {};
     exception InstallationFailure {};
     exception UnknownSoftwareDownloadTrackingObject {};
     exception SoftwareNotYetInstalled {};
     exception ActivationFailure {};
     exception ActivationCompleted {};
     exception ActivityCompleted {};
     exception ActivityInProgress {};
     exception InvalidSoftwareTrackingObject {};
     exception SoftwareTrackingObjectInUse {};

// Fin des définitions locales
```

```
valuetype DownloadMgrValueType: itut_x780::ManagedObjectValueType {

        public SoftwareDownloadTrackingObjectIdSeqType
ScheduledSoftwareDownloadTrackingObjectList; // GET
        public SoftwareDownloadTrackingObjectIdSeqType
OnDemandSoftwareDownloadTrackingObjectList; // GET

};


interface DownloadMgr : itut_x780::ManagedObject {

// Voir § 9.5.1.1 pour la description du comportement de cette opération

     SoftwareDownloadTrackingObjectIdType deliverDistSWGlobal (
          in FilenameSeqType softwareSet,
          in DCNAddressType softwareSourceAddr,
          in UserIdType userId,
          in PasswordType    password,
          in ManagedEntityIdSeqType deliverDistTargets)
          raises (CommFailure,
               UnrecognisedTarget,
               InsufficientMemory,
               SoftwareLoadHWMismatch,
               SourceUnreachable,
               UnknownSoftwareLoad,
               Timeout,
               AccessDenied,
               DeniedAccess);


// Voir § 9.5.1.2 pour la description du comportement de cette opération

     SoftwareDownloadTrackingObjectIdType deliverDistSWSpecific (
          in FilenameSeqType softwareSet,
          in DCNAddressType softwareSourceAddr,
          in UserIdType userId,
          in PasswordType    password,
          in TargetType  deliverDistTarget)
          raises (CommFailure,
               UnrecognisedTarget,
               InsufficientMemory,
               SoftwareLoadHWMismatch,
               SourceUnreachable,
               UnknownSoftwareLoad,
               Timeout,
               AccessDenied,
               DeniedAccess);

// Voir § 9.5.1.3 pour la description du comportement de cette opération

     void deleteSoftwareDownloadTrackingObject (
          in SoftwareDownloadTrackingObjectIdType id)
          raises (UnknownSoftwareDownloadTrackingObject, AccessDenied);

// Voir § 9.5.1.4  pour la description du comportement de cette opération

     void commit (
          in SoftwareDownloadTrackingObjectIdType id,
          in TargetType commitTarget)
          raises (InstallationFailure,
               UnknownSoftwareDownloadTrackingObject,
               AccessDenied,
```

```
                UnrecognisedTarget);

// Voir § 9.5.1.5 pour la description du comportement de cette opération

     void activate (
          in SoftwareDownloadTrackingObjectIdType id,
          in TargetType activateTarget)
          raises (UnknownSoftwareDownloadTrackingObject,
               SoftwareNotYetInstalled,
               ActivationFailure,
               AccessDenied,
               UnrecognisedTarget);

// Voir § 9.5.1.6 pour la description du comportement de cette opération

     void revert (
          in SoftwareDownloadTrackingObjectIdType id,
          in TargetType revertTarget)
          raises (UnknownSoftwareDownloadTrackingObject,
               SoftwareNotYetInstalled,
               ActivationFailure,
               AccessDenied,
               UnrecognisedTarget,
               InvalidSoftwareTrackingObject);

// Voir §  9.5.1.7 pour la description du comportement de cette opération

     DownloadStatusSeqType getStatus (
          in SoftwareDownloadTrackingObjectIdType id)
          raises (UnknownSoftwareDownloadTrackingObject,
               AccessDenied);

// Voir § 9.5.1.8 pour la description du comportement de cette opération

     SoftwareDownloadTrackingObjectIdType scheduleDeliverDist (
          in FilenameSeqType softwareSet,
          in DCNAddressType softwareSourceAddr,
          in UserIdType userId,
          in PasswordType password,
          in ManagedEntityIdSeqType deliverDistTargets,
          in GeneralizedTimeType deliverDistStartTime)
          raises (SoftwareLoadHWMismatch,
               UnknownScheduler,
               AccessDenied,
               InvalidStartTime);

// Voir § 9.5.1.9 pour la description du comportement de cette opération

     void scheduleCommit (
          in SoftwareDownloadTrackingObjectIdType
          deliverDistSoftwareDownloadTrackingObjectId,
          in GeneralizedTimeType commitStartTime)
          raises (UnknownSoftwareDownloadTrackingObject,
               UnknownScheduler,
               SoftwareNotYetInstalled,
               AccessDenied,
               InvalidStartTime);

// Voir § 9.5.1.10 pour la description du comportement de cette opération

     void scheduleActivate (
          in SoftwareDownloadTrackingObjectIdType id,
          in UserLabelType activateSchedulerName,
          in GeneralizedTimeType  activateStartTime)
```

```
                raises (UnknownSoftwareDownloadTrackingObject,
                     InvalidStartTime,
                     SoftwareNotYetInstalled,
                     AccessDenied,
                     InvalidScheduler );

     // Voir § 9.5.1.11 pour la description du comportement de cette opération

          void cancelScheduledSoftwareActivity (
               in SoftwareDownloadTrackingObjectIdType id)
               raises (UnknownSoftwareDownloadTrackingObject,
                     ActivityCompleted,
                     ActivityInProgress,
                     AccessDenied);

     // Voir § 9.5.1.12 pour la description du comportement de cette opération

          SoftwareDownloadTrackingObjectIdSeqType
     scheduledSoftwareDownloadTrackingObjectList ()
               raises (AccessDenied);

     // Voir § 9.5.1.13 pour la description du comportement de cette opération

          SoftwareDownloadTrackingObjectIdSeqType
     onDemandSoftwareDownloadTrackingObjectList ()
               raises (AccessDenied);


     }; // interface DownloadMgr



     interface VersionRepository : itut_x780::ManagedObject {

     // Voir § 9.5.2.1 pour la description du comportement de cette opération

          VersionsSeqType   retrieveVersions (
               in ManagedEntityIdType containingManagedEntityId)
               raises (CommFailure,
                     UnknownManagedEntity,
                     AccessDenied);

     // Voir § 9.5.2.2 pour la description du comportement de cette opération

          boolean validateNEVersion (
               in ManagedEntityIdType managedEntityId,
               in VersionType proposedSoftware)
               raises (UnknownNE, AccessDenied);

     // Voir § 9.5.2.3 pour la description du comportement de cette opération

          boolean validatePlugInVersion (
               in ManagedEntityIdType plugInUnitId,
               in VersionType proposedSoftware)
               raises (UnknownManagedEntity, AccessDenied);

}; // interface VersionRepository

}; // modules SoftwareDownload

}; // module q834_4

#endif
```

C.6 Q834EventPublisher.idl

```
#ifndef __Q834_4_EVENTPUBLISHING_DEFINED
#define __Q834_4_EVENTPUBLISHING_DEFINED
#include "Q834Common.idl"
#pragma prefix "itu.Int"


module  q834_4
{

module EventPublisher
{

// début des définitions issues d'autres fichiers en langage idl - données
// filtrables valeurs types


// de Q834Common


     typedef Q834Common::ManagedEntityIdType    ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::NameType NameType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::ProceduralStatusSetType ProceduralStatusSetType;
     typedef Q834Common::PerceivedSeverityType PerceivedSeverityType;
     typedef Q834Common::OperationalStateType OperationalStateType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::ProbableCauseType ProbableCauseType; // valeurs
définies dans Q834Common::ProbableCauseConst
     typedef Q834Common::NotificationIdentifierType NotificationIdentifierType;
     typedef Q834Common::NotificationIdentifierSeqType
NotificationIdentifierSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::EquipmentHolderFType EquipmentHolderFType;
     typedef Q834Common::PlugInUnitFType PlugInUnitFType;
     typedef Q834Common::UsageStateType UsageStateType;
     typedef Q834Common::ControlStatusSetType ControlStatusSetType;
     typedef Q834Common::SpecificProblemSetType SpecificProblemSetType;
     typedef Q834Common::BackedUpStatusType BackedUpStatusType;
     typedef Q834Common::AttributeChangeSetType AttributeChangeSetType;
     typedef Q834Common::SecurityAlarmCauseType SecurityAlarmCauseType;

// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales




     typedef NotificationIdentifierSeqType CorrelatedNotificationType;

     typedef AttributeChangeSetType StateChangeDefinitionType;


     struct ThresholdInfoType {
          MonitoredParameterType monitoredParameter;
          unsigned long long observedValue;
          unsigned long long thresholdValueLow;
```

```
        unsigned long long thresholdValueHigh;
    };

    /*
    Les valeurs pour monitoredParameter sont fournies dans
    Q834Common::MonitoringParameter.
    Actuellement, seuls les compteurs sont pris en charge pour la surveillance
    de la qualité de fonctionnement ou du trafic dans la technique des réseaux
    BPON: par conséquent, les valeurs observedValue, et thresholdValueHigh sont
    limitées à des nombres entiers non négatifs et la valeur du paramètre
    thresholdValueLow est 0.
    */

    typedef   unsigned shortEquipmentType;



    enum ServiceAffectingType {
        serviceAffecting,
        nonServiceAffecting,
        unableToDetermine
    };


// Fin des définitions locales de type de données

// Exceptions locales

// Fin des exceptions locales


    interface AlarmEventSupplier : itut_x780::ManagedObject {

        /*  Mappages d'événement structuré dans un en-tête fixe:
        domain_type est mis à la valeur "télécommunications",
        type_name est mis à la valeur "Alarm", et
        event_name est mis à la valeur d'une des chaînes constantes indiquées
        ci-dessous.

        const string communicationAlarm = "CommunicationAlarm";
        const string equipmentAlarm = "EquipmentAlarm";
        const string environmentalAlarm = "EnvironmentalAlarm";
        const string processingErrorAlarm = "ProcessingErrorAlarm";
        const string qualityOfServiceAlarm = "QualityOfServiceAlarm";

        /*
        Les noms des données filtrables pour remplir les champs d'événement
        filtrable dans l'événement structuré sont énumérés dans l'ordre ci-
        dessous.
        */

        const string alarmEmittingMEId = "AlarmEmittingMEId";
        const string eventTime = "EventTime";
        const string probableCause = "ProbableCause";
        const string specificProblems = "SpecificProblems";
        const string perceivedSeverity = "PerceivedSeverity";
        const string backUpStatus = "BackUpStatus";
        const string backUpManagedEntityId = "BackUpManagedEntityId";
        const string thresholdInfo = "ThresholdInfo";
        const string notificationIdentifier = "NotificationIdentifier";
        const string correlatedNotifications = "CorrelatedNotifications";
        const string stateChangeDefinition = "StateChangeDefinition";
        const string monitoredAttributes = "MonitoredAttributes";
```

```
        const string additionalText = "AdditionalText";
        const string serviceAffectingInd = "ServiceAffectingInd";

        /*
        Valeurs des données filtrables restantes.



        /* Noms des variables d'état et de statut pour StateChangeDefinition
        */

        const string administrativeState = "AdministrativeState";
        const string operationalState = "OperationalState";
        const string usageState = "UsageState";
        const string availabilityStatus = "AvailabilityStatus";
        const string proceduralStatus = "ProceduralStatus";
        const string controlStatus = "ControlStatus";
        const string alarmStatus = "AlarmStatus";


        /*
        Le mappage vers des données filtrables dans l'événement structuré est
        indiqué ci-dessous pour l'alarme de communication, l'alarme
        d'équipement, l'alarme d'erreur de traitement, l'alarme
        environnementale et l'alarme de qualité de service.

        {
        {"AlarmEmittingMEId", any (ManagedEntityIdType)},
        {"EventTime", any (GeneralizedTimeType)},
        {"ProbableCause", any (ProbableCauseType)},
        {"SpecificProblems", any (SpecificProblemSetType)},
        {"PerceivedSeverity", any (PerceivedSeverityType)},
        {"ServiceAffectingInd", any (ServiceAffectingType)},
        {"BackUpStatus", any (BackedUpStatusType)},
        {"BackUpManagedEntityId", any (ManagedEntityIdType)},
        {"ThresholdInfo", any (ThresholdInfoType)},
        {"NotificationIdentifier", any (NotificationIdentifierType)},
        {"CorrelatedNotifications", any (CorrelatedNotificationType)},
        {"StateChangeDefinition", any (StateChangeDefinitionType)},
        {"AdditionalText", any (string)}
        }

        */

};  // interface AlarmEventSupplier


interface SecurityEventSupplier : itut_x780::ManagedObject {

        /*   Mappages d'événement structuré dans un en-tête fixe:
        domain_type est mis à la valeur "télécommunications",
        type_name est mis à la valeur "SecurityEvent", et
        event_name est mis à la valeur d'une des chaînes constantes
        indiquées ci-dessous.
        */

        const string integrityViolation = "IntegrityViolation";
        const string operationalViolation = "OperationalViolation";
        const string physicalViolation = "PhysicalViolation";
        const string securityEventViolation = "SecurityEventViolation";
        const string timeDomainViolation = "TimeDomainViolation";
```

```
        const string eventEmittingMEId = "EventEmittingMEId";
        const string eventTime = "EventTime";
        const string securityAlarmCause = "SecurityAlarmCause";
        const string securityAlarmDetector = "SecurityAlarmDetector";
        const string serviceUser = "ServiceUser";
        const string serviceProvider = "ServiceProvider";
        const string securityEventNotificationIdentifier =
             "NotificationIdentifier";
        const string correlatedNotifications = "CorrelatedNotifications";
        const string additionalText = "AdditionalText";
        /*

        Le mappage vers des données filtrables dans l'événement structuré est
        indiqué ci-dessous pour les violations d'intégrité, opérationnelles,
        physiques, d'événement de sécurité et de domaine temporel.


        {
        {"EventEmittingMEId", any (ManagedEntityIdType)},
        {"EventTime", any (GeneralizedTimeType)},
        {"SecurityAlarmCause", any (SecurityAlarmCauseType)},
        {"SecurityAlarmDetector", any (ManagedEntityIdType)},
        {"ServiceUser", any (ManagedEntityIdType)},
        {"ServiceProvider", any (ManagedEntityIdType)},
        {"NotificationIdentifier", any (NotificationIdentifierType)},
        {"CorrelatedNotifications", any (CorrelatedNotificationType)},
        {"AdditionalText", any (string)}
        }

        */

    }; // interface SecurityEventSupplier


    interface DiscoveryEventSupplier : itut_x780::ManagedObject {

        /*  Mappages d'événement structuré dans un en-tête fixe:
        domain_type est mis à la valeur "télécommunications",
        type_name est mis à la valeur "DiscoveryEvent", et
        event_name est mis à la valeur d'une des chaînes constantes
        ci-dessous. Seules les modifications apportées à un équipement
        installé sont déclarées au moyen de cette interface.
        */

        const string managedEntityCreation = "ManagedEntityCreation";
        const string managedEntityDeletion = "ManagedEntityDeletion";


        const string managedEntityType = "ManagedEntityType";
        const string managedEntityAttributeValues =
        "ManagedEntityAttributeValues"; const string
        discoveryNotificationIdentifier = "NotificationIdentifier";
        const string correlatedNotifications = "CorrelatedNotifications";
        const string additionalInformation = "AdditionalInformation";

        // Les items suivants sont des types d'équipement qui sont
        // découverts par le système de gestion du fournisseur et qui sont
        // automatiquement révélés au système OMS. La structure de données
        // transmise par la notification est définie dans le fichier
        // Q834Common.idl et le mappage ci-dessous s'y
        // réfère par MEstruct. Plus spécifiquement, voici la liste des
        // structures de données actuellement prises en charge: OLTType,
        // ONUType, ONTType, NTType, equipmentHolderType et PlugInUnitFType.
        const unsigned short OLT_NE = 1;
```

```
        const unsigned short ONT_NE = 2;
        const unsigned short ONU_NE = 3;
        const unsigned short NT_NE = 4;
        const unsigned short EQUIPMENT_HOLDER_F = 5;
        const unsigned short PLUG_IN_UNIT_F = 6;


        /*
        Le mappage vers des données filtrables dans l'événement structuré est
        indiqué ci-dessous pour un événement de découverte qui implique la
        création d'une entité gérée.


        {
        {"ManagedEntityType", any (EquipmentType)},
        {"EventTime", any (GeneralizedTimeType)},
        {"ManagedEntityAttributeValues", any (MEstruct)},
        {"NotificationIdentifier", any (NotificationIdentifierType)},
        {"CorrelatedNotifications", any (CorrelatedNotificationType)},
        {"AdditionalText", any (string)}
        }

        */

    }; // interface DiscoveryEventSupplier

}; // module EventPublisher

}; // module q834_4

#endif
```

C.7 Q834MIBTransfer.idl

```
#ifndef __Q834_4_MIBTRANSFER_DEFINED
#define __Q834_4_MIBTRANSFER_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module MIBTransfer {
// Début des définitions issues d'autres fichiers en langage idl


// de Q834Common

    typedef Q834Common::UserLabelType UserLabelType;
    typedef Q834Common::DCNAddressType  DCNAddressType;
    typedef Q834Common::FilenameType  FilenameType;
    typedef Q834Common::StatusAttributeSeqType StatusAttributeSeqType;
    typedef Q834Common::TransferTrackingObjectIdType
    TransferTrackingObjectIdType;
    typedef Q834Common::UserIdType UserIdType;
    typedef Q834Common::PasswordType PasswordType;
    typedef Q834Common::ManagedEntityIdType    ManagedEntityIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define EquipmentFailure Q834Common::EquipmentFailure
#define UnknownNE Q834Common::UnknownNE
#define UnknownScheduler Q834Common::UnknownScheduler
```

```
#define InvalidScheduler Q834Common::InvalidScheduler
#define UnknownDestinationServer Q834Common::UnknownDestinationServer
#define FileExists Q834Common::FileExists
#define CannotCreateFile Q834Common::CannotCreateFile
#define DeniedAccess Q834Common::DeniedAccess

// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales

     typedef sequence<TransferTrackingObjectIdType>
TransferTrackingObjectIdSeqType;

// Exceptions locales

     exception UnknownBackupProcess {};
     exception UnknownSourceServer {};
     exception UnknownRestoreProcess {};
     exception SoftwareLoadHardwareMismatch {};


// Fin des définitions locales

     valuetype MIBMoverValueType: itut_x780::ManagedObjectValueType {

          public TransferTrackingObjectIdSeqType transferTrackingObjectIdList;
// GET

     };

     interface MIBMover : itut_x780::ManagedObject {

     // Voir § 9.7.1.1 pour la description du comportement de cette opération

          TransferTrackingObjectIdType startBackup (
               in ManagedEntityIdType nEManagedEntityId, // OLT
               in DCNAddressType destinationServerAddr,
               in UserIdType userId,
               in PasswordType    password,
               in FilenameType destinationFile,
               in boolean overwriteExistingFile)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownDestinationServer,
                    CommFailure,
                    EquipmentFailure,
                    DeniedAccess);

     // Voir § 9.7.1.2 pour la description du comportement de cette opération

          StatusAttributeSeqType getBackupStatus (
               in TransferTrackingObjectIdType id)
               raises (AccessDenied,
                    UnknownBackupProcess);


     // Voir § 9.7.1.3 pour la description du comportement de cette opération

          TransferTrackingObjectIdType scheduleBackup (
               in ManagedEntityIdType nEManagedEntityId, // OLT
               in UserIdType userId,
               in PasswordType    password,
               in UserLabelType schedulerName,
               in DCNAddressType destinationServerAddr,
```

```
          in FilenameType destinationFile,
          in boolean overwriteExistingFile)
          raises (AccessDenied,
               UnknownNE,
               UnknownScheduler,
               UnknownDestinationServer,
               InvalidScheduler);

// Voir § 9.7.1.4 pour la description du comportement de cette opération

        void modifyBackupSchedule(
          in TransferTrackingObjectIdType id,
          in UserLabelType newSchedulerName)
          raises (AccessDenied,
               UnknownBackupProcess,
               UnknownScheduler,
               InvalidScheduler);

// Voir § 9.7.1.5 pour la description du comportement de cette opération

     void cancelScheduledBackup (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownBackupProcess);

// Voir § 9.7.1.6 pour la description du comportement de cette opération

     void abortBackup (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownBackupProcess,
               CommFailure,
               EquipmentFailure);

// Voir § 9.7.1.7 pour la description du comportement de cette opération

     TransferTrackingObjectIdType startRestore (
          in ManagedEntityIdType nEManagedEntityId,     // OLT
          in DCNAddressType sourceServerAddr,
          in UserIdType userId,
          in PasswordType    password,
          in FilenameType sourceFile)
          raises (AccessDenied,
               UnknownNE,
               UnknownSourceServer,
               CommFailure,
               EquipmentFailure,
               DeniedAccess,
               SoftwareLoadHardwareMismatch );

// Voir § 9.7.1.8 pour la description du comportement de cette opération

     StatusAttributeSeqType getRestoreStatus (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownRestoreProcess);


// Voir § 9.7.1.9 pour la description du comportement de cette opération

     TransferTrackingObjectIdSeqType transferTrackingObjectIdListGet ()
          raises (AccessDenied);

}; // interface MIBMover
```

```
}; // module MIBTransfer

}; // module q834_4

#endif
```

C.8 Q834PerformanceManager.idl

```
#ifndef __Q834_4_PERFORMANCEMANAGER_DEFINED
#define __Q834_4_PERFORMANCEMANAGER_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module PerformanceManager {

     // début des définitions issues d'autres fichiers en langage idl

     // de Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::RecordSeqType RecordSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::MonitoringKindType MonitoringKindType;
     typedef Q834Common::RecordKindType RecordKindType;


     #define AccessDenied Q834Common::AccessDenied
     #define UnknownNE Q834Common::UnknownNE
     #define UnknownManagedEntity Q834Common::UnknownManagedEntity
     #define UnknownProfiles Q834Common::UnknownProfiles
     #define UnknownServiceInstance Q834Common::UnknownServiceInstance
     #define UnknownScheduler Q834Common::UnknownScheduler
     #define CommFailure Q834Common::CommFailure
     #define InvalidScheduler Q834Common::InvalidScheduler
     #define EquipmentFailure Q834Common::EquipmentFailure
     #define ProfileSuspended Q834Common::ProfileSuspended

     // Fin des définitions issues d'autres fichiers en langage idl

     // Types de données locales


     struct  MonitoringPointKindAndThresholdsType {
          MonitoringKindType monitoringType;
          NameType  thresholdDataProfileName;
     };
     /*
     NOTE – Les Recommandations UIT-T Q.834.1 et Q.834.2 décrivent les relations
     entre les données de seuil et les types de point de surveillance
     spécifiques.
     */

     typedef sequence<MonitoringPointKindAndThresholdsType> ThresholdsSeqType;
```

```
     struct MonitoringPointAndThresholdsType {
          ManagedEntityIdType monitoringPoint;
          NameType  thresholdDataProfileName;
     };

     typedef sequence<MonitoringPointAndThresholdsType>
MonitoringPointThresholdsSeqType;

     typedef sequence<MonitoredParameterType>  MonitoredParameterSeqType;

     typedef sequence<RecordSeqType> RecordsSeqType; // Implicitement groupés
                                                     //  par type d'article.

     typedef RecordKindType  HistoryDataType; // valeurs définies dans
                                              // Q834Common::RecordSetType

     typedef ManagedEntityIdSeqType   MonitoringPointSeqType;

     struct SWPValueType {
          ManagedEntityIdType monitoringPoint;
          short          totConsecutiveIntvls;
          short          persistenceMinimum;
     };

     typedef sequence< SWPValueType > SWPValueSeqType;


     struct  ParameterSettingType  {
          MonitoredParameterType       monitoredParameter;
          short          totConsecutiveIntvls;
          short          persistenceMinimum;
     };

     typedef sequence<ParameterSettingType> ParameterSettingSeqType;

     // Exceptions locales

     exception UnknownParameters {
          MonitoredParameterSeqType monitoredParameterList;
     };
     exception IntervalCountTooLarge {};
     exception  UnknownMonitoringPointTypes {
          MonitoringPointSeqType monitoringPointKindList;
     };
     exception InvalidAssociation {};
     exception CollectionPeriodPast {};
     exception CollectionLimitation {};
     exception UnknownHistoryDataType {};

     // Fin des définitions locales


     interface ImpairmentPersistence : itut_x780::ManagedObject {

          // Voir § 9.8.1.1 pour la description du comportement de cette
          //  opération

          void setSlidingWindowParameters (
               in ManagedEntityIdType nEManagedEntityId,
               in MonitoredParameterSeqType monitoredParameterList,
               in short totConsecutiveIntvls,
               in short persistenceMinimum,
               in boolean sysScopeInd)
```

```
     raises (UnknownNE,
          UnknownParameters,
          IntervalCountTooLarge,
          AccessDenied,
          CommFailure);

 // Voir § 9.8.1.2 pour la description du comportement de cette
// opération

 void setSpecificSlidingWindowParameters (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in MonitoredParameterSeqType monitoredParameterList,
     in short totConsecutiveIntvls,
     in short persistenceMinimum,
     in boolean allowGlobalOverwrite)
     raises (UnknownNE,
          UnknownParameters,
          IntervalCountTooLarge,
          AccessDenied,
          UnknownManagedEntity,
          CommFailure,
          EquipmentFailure);

 // Voir § 9.8.1.3 pour la description du comportement de cette
 // opération

 SWPValueSeqType getSpecificSlidingWindowParameters (
     in ManagedEntityIdType nEManagedEntityId,
     in MonitoredParameterType monitoredParameter)
     raises (UnknownNE,
          UnknownParameters,
          CommFailure);

 // Voir § 9.8.1.4 pour la description du comportement de cette
 // opération

 void setThreshold (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in NameType thresholdDataProfileName)
     raises (UnknownNE,
          AccessDenied,
          UnknownManagedEntity,
          UnknownProfiles,
          InvalidAssociation,
          CommFailure,
          ProfileSuspended);

 // Voir § 9.8.1.5 pour la description du comportement de cette
 // opération

 void setThresholds (
     in ManagedEntityIdType nEManagedEntityId,
     in boolean sysScopeInd,
     in ThresholdsSeqType thresholdsList)
     raises (UnknownNE,
          UnknownProfiles,
          AccessDenied,
          UnknownMonitoringPointTypes,
          InvalidAssociation,
          CommFailure,
          ProfileSuspended);
```

```
     // Voir § 9.8.1.6 pour la description du comportement de cette
     // opération

     MonitoringPointThresholdsSeqType getThresholdValues (
          in ManagedEntityIdType nEManagedEntityId,
          in MonitoringKindType monitoringType)
          raises (UnknownNE,
               UnknownMonitoringPointTypes,
               CommFailure);

     // Voir § 9.8.1.7 pour la description du comportement de cette
     // opération

     ThresholdsSeqType getSystemThresholdsSetting(
          in ManagedEntityIdType nEManagedEntityId)
          raises (AccessDenied, UnknownManagedEntity);

     // Voir § 9.8.1.8 pour la description du comportement de cette
     // opération

     ParameterSettingSeqType getSystemSWSettings(
          in ManagedEntityIdType nEManagedEntityId)
          raises (AccessDenied, UnknownManagedEntity);

}; // interface ImpairmentPersistence


interface ReportController : itut_x780::ManagedObject {

     // Voir § 9.8.2.1 pour la description du comportement de cette
     // opération

     void addCustomerMonitoringReporting  (
          in ManagedEntityIdType nEManagedEntityId,
          in ServiceInstanceIdType serviceInstanceId,
          in ManagedEntityIdType monitoringPoint,
          in GeneralizedTimeType stopTime,
          in HistoryDataType historyData,
          in short granularityPeriod) //in minutes
          raises (UnknownServiceInstance,
               AccessDenied,
               UnknownNE,
               UnknownManagedEntity,
               CollectionPeriodPast,
               CollectionLimitation,
               InvalidAssociation,
               UnknownHistoryDataType,
               CommFailure);

     // Voir § 9.8.2.2 pour la description du comportement de cette
     // opération

     void removeCustomerMonitoringReporting  (
          in ServiceInstanceIdType serviceInstanceId)
          raises (UnknownServiceInstance,
               AccessDenied,
               CollectionPeriodPast,
               CommFailure);

     // Voir § 9.8.2.3 pour la description du comportement de cette
     // opération

     RecordsSeqType selectByServiceInstance (
          in ServiceInstanceIdType serviceInstanceId,
```

```
     in GeneralizedTimeType intervalStartTime,
     in GeneralizedTimeType intervalEndTime)
     raises (UnknownServiceInstance,
          AccessDenied );

// Voir § 9.8.2.4 pour la description du comportement de cette
// opération

MonitoringPointSeqType displayActiveReporting (
     in ServiceInstanceIdType serviceInstanceId)
     raises (UnknownServiceInstance,
          AccessDenied);

// Voir § 9.8.2.5 pour la description du comportement de cette
// opération

void addNewMonitoringReporting (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in GeneralizedTimeType stopTime,
     in HistoryDataType historyData,
     in short granularityPeriod) //in minutes
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionPeriodPast,
          CollectionLimitation,
          InvalidAssociation,
          UnknownHistoryDataType,
          CommFailure);

// Voir § 9.8.2.6 pour la description du comportement de cette
// opération

RecordsSeqType selectByMonitoringPoint (
     in ManagedEntityIdType monitoringPoint,
     in GeneralizedTimeType intervalStartTime,
     in GeneralizedTimeType intervalEndTime)
     raises (UnknownManagedEntity,
          AccessDenied);


// Voir § 9.8.2.7 pour la description du comportement de cette
// opération

void createReportingSchedule (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in HistoryDataType historyData,
     in ServiceInstanceIdType serviceInstance,
     in short granularityPeriod,  //in minutes
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionLimitation,
          UnknownScheduler,
          InvalidAssociation,
          UnknownHistoryDataType,
          InvalidScheduler);

// Voir § 9.8.2.8 pour la description du comportement de cette
// opération
```

```
          void modifyReportingSchedule (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in UserLabelType newSchedulerName)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownManagedEntity,
                    CollectionLimitation,
                    UnknownScheduler,
                    InvalidAssociation,
                    UnknownHistoryDataType,
                    InvalidScheduler);

          // Voir § 9.8.2.9 pour la description du comportement de cette
          // opération

          void cancelReportingSchedule (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in UserLabelType schedulerName)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownManagedEntity,
                    UnknownScheduler,
                    InvalidAssociation,
                    UnknownHistoryDataType);

     }; // interface ReportController

}; // module PerformanceManager

}; // module q834_4

#endif
```

C.9 Q834ProfileManager.idl

```
#ifndef __Q834_4_PROFILEMANAGER_DEFINED
#define __Q834_4_PROFILEMANAGER_DEFINED
#include "Q834Common.idl"
#pragma prefix "itu.Int"


module  q834_4
{

module ProfileManager
{

     // début des définitions issues d'autres fichiers en langage idl - types de
     // valeur de données filtrables
     typedef Q834Common::NameType NameType;
     typedef Q834Common::NotificationIdentifierType NotificationIdentifierType;
     typedef Q834Common::PerceivedSeverityType PerceivedSeverityType;
     typedef Q834Common::ProbableCauseType ProbableCauseType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::ServiceCategoryType ServiceCategoryType;
     typedef Q834Common::ConformanceDefType ConformanceDefType;
     typedef Q834Common::ATMOverbookingFactorType ATMOverbookingFactorType;
     typedef Q834Common::MonitoringKindType MonitoringKindType;
```

```
#define UnknownProfiles Q834Common::UnknownProfiles
#define AccessDenied Q834Common::AccessDenied

     // Fin des définitions issues d'autres fichiers en langage idl

     // Types de données locales

     typedef unsigned short  ProfileKindType;

     struct  ProfileInfoType {
          ProfileKindType    profileKind;
          any  attributeValuestruct;
     };

     /*
     Ensuite, les structures (et les séquences de structures) sont définies
     comme ayant les valeurs d'attribut pour les objets de profil nouvellement
     créés. Chacune de ces structures sera ultérieurement désignée comme
     ProfileStruct dans la spécification de l'interface ProfileEventConsumer.
     */

     enum  SubType  {
          NULL,
          VoicebandBasedOn64kbps,
          SynchronousCircuitEmulation,
          AsynchronousCircuitEmulation,
          HighQualityAudio,
          Video
     };

     enum CBRRateType {
          br44736kbps,
          br1544kbps,
          br2048kbps,
          brE3kbps,
          br64kbps,
          br2x64kbps,
          br3x64kbps,
          br4x64kbps,
          br5x64kbps,
          br6x64kbps,
          br7x64kbps,
          br8x64kbps,
          br9x64kbps,
          br10x64kbps,
          br11x64kbps,
          br12x64kbps,
          br13x64kbps,
          br14x64kbps,
          br15x64kbps,
          br16x64kbps,
          br17x64kbps,
          br18x64kbps,
          br19x64kbps,
          br20x64kbps,
          br21x64kbps,
          br22x64kbps,
          br23x64kbps,
          br24x64kbps,
          br25x64kbps,
          br26x64kbps,
          br27x64kbps,
          br28x64kbps,
          br29x64kbps,
```

```
     br30x64kbps,
     br31x64kbps
};

enum ClockRecoveryType {
     PhysInterface,
     SRTS,
     AdaptiveClock,
     LocalOsc
};

enum ForwardErrorCorrectionType {
     NoFEC,
     FECforLossSensitiveSigTransport,
     FECforDelaySensSigTransport
};

typedef boolean  StructuredDataTransferType; //TRUE = STD a été choisi
typedef long long  PartiallyFilledCellsType;
typedef long long  CellLossIntegrationPeriodType;

struct   AAL1ProfileType {
     SubType  aAL1subtype;
     CBRRateType   cBRRate;
     ClockRecoveryType  clockRecovery;
     ForwardErrorCorrectionType  forwardErrorCorrection;
     StructuredDataTransferType  structuredDataTransfer;
     PartiallyFilledCellsType    partiallyFilledCells;
}; // Structure de profil pour ce type de profil = 1


typedef stringDefaultSSCSParameterProfile1PtrType;

typedef stringDefaultSSCSParameterProfile2PtrType;

struct   AAL2ProfileType {
     DefaultSSCSParameterProfile1PtrType defaultSSCSParameterProfile1Ptr;
     DefaultSSCSParameterProfile2PtrType defaultSSCSParameterProfile2Ptr;
}; // Structure de profil pour ce type de profil = 2


struct   MaxCPCSSDUSizeType  {
     long long forwardDirectionCPCS_SDU;
     long long backwardsDirectionCPCS_SDU;
};

enum AALModeType {
     MessageAssured,
     MessageUnassured,
     StreamingAssured,
     StreamingUnassured
};

enum SSCSType {
     NoSSC,
     DataSSCSonSSCOPAssured,
     DataSSCSonSSCOPNotAssured,
     FrameRelaySSCS
};

struct   AAL5ProfileType {
     // ManagedEntityIdType mEId; ??
     MaxCPCSSDUSizeType maxCPCSSDUSize;
     AALModeType   aALMode;
```

```
        SSCSType  sSCS;
    }; // Structure de profil pour ce type de profil = 4


    enum AppIdType {
        LES_CAS_noELCP,
        LES_PSTN_noELCP,
        LES_PSTN_ELCP,
        LES_DSS1forBRI_noELCP,
        LES_DSS1forBRI_ELCP,
        LES_CASforPOTS_DSS1forBRI_noELCP,
        LES_PSTNforPOTS_DSS1forBRI_noELCP,
        LES_PSTNforPOTS_DSS1forBRI_ELCP,
        LES_otherCCS,
        UnspecifiedLES
    };

    typedef long long  MaxNumChannelsType;
    typedef long long  MinimumChanIdType;
    typedef long long  MaximumChanIdType;
    typedef long long  MaxCPS_SDULengthType;
    typedef long long  TimerCULengthType;

    struct   AAL2PVCProfileType {
        MaxNumChannelsType maxNumChannels;
        MinimumChanIdType  minimumChanId;
        MaximumChanIdType  maximumChanId;
        MaxCPS_SDULengthType    maxCPS_SDULength;
        TimerCULengthType  timerCULength;
    }; // Structure de profil pour ce type de profil = 3

    struct    AlarmSeverityAssignProfileType {
        string eventName; //As defined in AlarmEventSupplier
        ProbableCauseType  probableCauseValue; //Ditto as above
        PerceivedSeverityType   serviceAffectingSeverity;
        PerceivedSeverityType   nonServiceAffectingSeverity;
    }; // Structure de profil pour ce type de profil = 5

    typedef sequence<AlarmSeverityAssignProfileType>
AlarmSeverityAssignProfileSeqType;
    typedef long long  LocalMaxNumVPCSupportedType;
    typedef long long LocalMaxNumVCCSupportedType;
    typedef short LocalMaxNumVPIBitsType;
    typedef short LocalMaxNumVCIBitsType;
    typedef long long  TotalEgressBandwidthType;
    typedef long long  TotalIngressBandwidthType;
    typedef boolean    UPCNPCIndicatorType; //TRUE = la surveillance est
    activée

    struct   ATMNetworkAccessProfileType {
        LocalMaxNumVPCSupportedType localMaxNumVPCSupported;
        LocalMaxNumVCCSupportedType localMaxNumVCCSupported;
        LocalMaxNumVPIBitsType  localMaxNumVPIBits;
        LocalMaxNumVCIBitsType  localMaxNumVCIBits;
        TotalEgressBandwidthType    totalEgressBandwidth;
        TotalIngressBandwidthType   totalIngressBandwidth;
        UPCNPCIndicatorType    uPCNPCIndicator;
    }; // Structure de profil pour ce type de profil = 6


    typedef short LANType;
    typedef short EncapsulationProtocolType;
    typedef short PIDType;
```

```
struct   BridgedLANServiceProfileValues {
     LANType  lAN;
     EncapsulationProtocolType   encapsulationProtocol;
     PIDType  pID;
}; // Structure de profil pour ce type de profil = 7


typedef long long  BufferedCDVToleranceType;

enum CASType {
     basic,
     e1Cas,
     SfCas,
     ds1EsfCas,
     j2Cas
};

typedef long long  CableLengthType;

struct   CESServiceProfileValues {
     BufferedCDVToleranceType bufferedCDVTolerance;
     CASType  cAS;
}; // Structure de profil pour ce type de profil = 8


enum DS1FramingType {
     SuperFrame,
     ExtendedSuperFrame,
     Unframed
};

enum DS1EncodingType {
     AMI,
     B8ZS
};

enum  LoopbackCodeType {
     SmartJack,
     SmartJack_ONTInband,
     COInband
};

typedef boolean SupplyPowerIndType;

struct   DS1ProfileValues {
     DS1FramingType     dS1Framing;
     DS1EncodingType dS1Encoding;
     LoopbackCodeType   loopbackCode;
     SupplyPowerIndType supplyPowerInd;
     CableLengthType    cableLength;
}; // Structure de profil pour ce type de profil = 9


enum DS3ApplicationType {
     CBitParity,
     M23
};

enum DS3EncodingType {
     B3ZS,
     CCHAN
};

struct   DS3ProfileType {
```

```
     DS3ApplicationType dS3Application;
     DS3EncodingType    dS3Encoding;
     CableLengthType    cableLength;
}; // Structure de profil pour ce type de profil = 10

typedef    boolean DuplexType; //TRUE = bilatéral, FALSE = alternat
typedef boolean AutoDetectionIndType;

enum DataRateType {
     TenBT,
     HundredBT,
     ThousandBT,
     otherrate
};

typedef long long   MaxFrameSizeType;  //Fixe pour Ethernet

typedef  boolean DTEDCEType;  // TRUE = réglage ETTD; FALSE = réglage ETCD.

struct   EthernetProfileValues {
     DuplexType    duplex;
     AutoDetectionIndType    autoDetectionInd;
     DataRateType  dataRate;
     MaxFrameSizeType   maxFrameSize;
     DTEDCEType    dTEDCE;
}; // Structure de profil pour ce type de profil = 11


enum  T303Type {
     m700,
     m1200,
     m1700,
     m2200,
     m2700,
     m3200,
     m3700,
     m4200,
     m4700
};

enum T396Type {
     ms700,
     ms1700,
     ms2700,
     ms3700,
     ms4700,
     ms5700,
     ms6700,
     ms7700,
     ms8700,
     ms9700,
     ms10700,
     ms11700,
     ms12700,
     ms13700,
     ms14700
};

struct   IDLCCallProcessingProfileType {
     T303Type  t303;
     T396Type  t396;
}; // Structure de profil pour ce type de profil = 12

typedef boolean    ELCPIndType;
```

```
enum POTSSignallingType {
     PSTN,
     ChAS,
     CCS,
     UNKNOWN
};

enum  BRISignallingType {
     DSS1,
     OtherCCS
};

typedef long long MaxNumCIDsType;
typedef long long MaxPacketLengthType;
struct  ChannelWithSSCSPtrType {
     long channelIndex;
     boolean   ptr1orPtr2;
};

typedef sequence<ChannelWithSSCSPtrType>   ChanAndSSCSParaPtrSeqType;

struct  LESProfileType {
     ELCPIndType   eLCPInd;
     POTSSignallingType pOTSSignalling;
     BRISignallingType  bRISignalling;
     MaxNumCIDsType     maxNumCIDs;
     MaxPacketLengthType     maxPacketLength;
     ChanAndSSCSParaPtrSeqType    chanAndSSCSParaPtrSeq;
}; // Structure de profil pour ce type de profil = 13

typedef boolean    SpanningTreeIndType;
typedef short BridgePriorityType;
typedef short MaxAgeType;
typedef short HelloTimeType;
typedef short ForwardDelayType;

struct  MACBridgeServiceProfileType {
     SpanningTreeIndType     spanningTreeInd;
     BridgePriorityType      bridgePriority;
     MaxAgeType    maxAge;
     HelloTimeType helloTime;
     ForwardDelayType   forwardDelay;
}; // Structure de profil pour ce type de profil = 14

typedef long long  SegmentLengthType;
typedef short RASTimerType;
typedef long long  MaxSSSARSDULengthType;
typedef boolean    SSTEDIndType;
typedef boolean    SSADTIndType;

struct    SSCSParameterProfile1Type {
     SegmentLengthType  segmentLength;
     RASTimerType  rASTimer;
     MaxSSSARSDULengthType maxSSSARSDULength;
     SSTEDIndType  sSTEDInd;
     SSADTIndType  sSADTInd;
}; // Structure de profil pour ce type de profil = 15

enum ServiceCatType {
     Audio,
     Multirate,
     UNKNCategory
};
```

```
enum EncSrcType {
     ITUT,
     ATMForum,
     Proprietary
};

typedef short  EncProfileIndexType;
typedef boolean    AudioServIndType;
typedef short  PCMEncType;
typedef boolean    CMDataIndType;
typedef short  CMMultiplierNumType;
typedef boolean    FMDataIndType;
typedef long long  FMMaxFrameLenType;
typedef boolean    CASIndType;
typedef boolean    DTMFIndType;
typedef boolean    MFR1IndType;
typedef boolean    MFR2IndType;
typedef boolean    RateControlIndType;
typedef boolean    SynchChangeIndType;
typedef boolean    FaxDemodIndType;

struct   SSCSParameterProfile2Type {
     ServiceCatType     serviceCat;
     EncSrcType    encSrc;
     EncProfileIndexType     encProfileIndex;
     AudioServIndType   audioServInd;
     PCMEncType    pCMEnc;
     CMDataIndType cMDataInd;
     CMMultiplierNumType     cMMultiplierNum;
     FMDataIndType fMDataInd;
     FMMaxFrameLenType  fMMaxFrameLen;
     CASIndType    cASInd;
     DTMFIndType   dTMFInd;
     MFR1IndType   mFR1Ind;
     MFR2IndType   mFR2Ind;
     RateControlIndType rateControlInd;
     SynchChangeIndType      synchChangeInd;
     FaxDemodIndType    faxDemodInd;
}; // Structure de profil pour ce type de profil = 16


typedef long long  PCRIngressType;
typedef long long PCREgressType;
typedef long long  CDVTPCRIngressType;
typedef long long  CDVTPCREgressType;
typedef long long CDVTSCRIngressType;
typedef long long CDVTSCREgressType;
typedef long long SCRIngressType;
typedef long long  SCREgressType;
typedef long long  MaxBSIngressType;
typedef long long  MaxBSEgressType;
typedef long long  MFSIngressType;
typedef long long MFSEgressType;

struct   TrafficDescriptorProfileType {
     ServiceCategoryType     serviceCategory;
     ConformanceDefType conformanceDef;
     PCRIngressTypepCRIngress;
     PCREgressType pCREgress;
     CDVTPCRIngressType cDVTPCRIngress;
     CDVTPCREgressType  cDVTPCREgress;
     CDVTSCRIngressType cDVTSCRIngress;
```

```
     CDVTSCREgressType  cDVTSCREgress;
     SCRIngressTypesCRIngress;
     SCREgressType sCREgress;
     MaxBSIngressType   maxBSIngress;
     MaxBSEgressType    maxBSEgress;
     MFSIngressTypemFSIngress;
     MFSEgressType mFSEgress;
}; // Structure de profil pour ce type de profil = 17


typedef  string LoopbackLocCodeType;

struct   UNIProfileType {
     LocalMaxNumVPCSupportedType localMaxNumVPCSupported;
     LocalMaxNumVCCSupportedType localMaxNumVCCSupported;
     LocalMaxNumVPIBitsType  localMaxNumVPIBits;
     LocalMaxNumVCIBitsType  localMaxNumVCIBits;
     LoopbackLocCodeType     loopbackLocCode;
}; // Structure de profil pour ce type de profil = 18


enum AnnouncementType {
     Silence,
     ReorderTone,
     FastBusy,
     VoiceAnnouncement,
     OtherAnnouncementType,
     UNKNAnnouncementType
};

enum TimingReferenceType {
     NetworkTimingReference,
     AdaptiveVoice,
     FreeRun,
     OtherTimingReference
};

typedef boolean    EchoCancellationIndType;

struct   VoiceServiceProfileAAL1Type {
     AnnouncementType   announcement;
     TimingReferenceType     timingReference;
     EchoCancellationIndType echoCancellationInd;
}; // Structure de profil pour ce type de profil = 19


typedef long long  JitterTargetType;
typedef long long  JitterBufferMaxType;

struct   VoiceServiceProfileAAL2Type {
     AnnouncementType   announcement;
     TimingReferenceType     timingReference;
     JitterTargetType   jitterTarget;
     JitterBufferMaxType     jitterBufferMax;
     EchoCancellationIndType echoCancellationInd;
}; // Structure de profil pour ce type de profil = 20

struct   ThresholdDataComponentType {
     MonitoredParameterType monitoredParameter; // Valeurs définies dans
     Q834Common::MonitoringParameter
     unsigned long long thresholdValue;
};

typedef sequence<ThresholdDataComponentType>     ThresholdDataSeqType;
```

```
struct ThresholdDataProfileType {
     MonitoringKindType monitoringType;
     ThresholdDataSeqType thresholdValues;
}; // Structure de profil pour ce type de profil = 21

typedef sequence<ATMOverbookingFactorType> ATMOverbookingProfileType;
// Structure de profil pour ce type de profil = 22

// Exceptions locales
exception ProfileInUse {};
exception DuplicateProfileName {};

// Fin des définitions locales


interface ProfileConsumer : itut_x780::ManagedObject {
     /*
     Mappages d'événement structuré dans un en-tête fixe:
     domain_type est mis à la valeur "télécommunications",
     type_name est mis à la valeur "ProfileEvent", et
     event_name est mis à la chaîne constante indiquée
     ci-dessous. Seuls les nouveaux profils sont annoncés
     par cette interface.
     */

     const string profileCreation = "ProfileCreation";

     /*
     Identification des items restants dans les données filtrables
     d'événement structuré.
     */

     const string profileName = "ProfileName";
     const string profileType = "ProfileType";
     const string profileAttributeValues = "ProfileAttributeValues";

     /*
     Les valeurs identifiant les types de profil utilisés sont indiquées
     ci-dessous.
     */

     const  unsigned short   AAL1Profile = 1;
     const  unsigned short   AAL2Profile = 2;
     const  unsigned short   AAL2PVCProfile = 3;
     const  unsigned short   AAL5Profile = 4;
     const  unsigned short   AlarmSeverityAssignmentProfile = 5;
     const  unsigned short   ATMNetworkAccessProfile = 6;
     const  unsigned short   BridgedLANServiceProfile = 7;
     const  unsigned short   CESServiceProfile = 8;
     const  unsigned short   DS1Profile = 9;
     const  unsigned short   DS3Profile = 10;
     const  unsigned short   EthernetProfile = 11;
     const  unsigned short   IDLCCallProcessingProfile = 12;
     const  unsigned short   LESProfile = 13;
     const  unsigned short   MACBridgeServiceProfile = 14;
     const  unsigned short   SSCSParameterProfile1 = 15;
     const  unsigned short   SSCSParameterProfile2 = 16;
     const  unsigned short   TrafficDescriptorProfile = 17;
     const  unsigned short   UNIProfile = 18;
     const  unsigned short   VoiceServiceProfileAAL1 = 19;
     const  unsigned short   VoiceServiceProfileAAL2 = 20;
     const  unsigned short   ThresholdData = 21;
     const  unsigned short   ATMOverbookingFactorProfile = 22;
```

```
     /*
     Le mappage vers des données filtrables dans l'événement structuré est
     fourni pour un événement de consommateur qui implique la création d'un
     objet de profil.

     {
     {"ProfileName", any (NameType)},
     {"ProfileType", any (ProfileKindType)},
     {"EventTime", any (GeneralizedTimeType)},
     {"ProfileAttributeValues", any (ProfileStruct)},
     {"NotificationIdentifier", any (NotificationIdentifierType)}
     }

     */


}; // interface ProfileConsumer


interface ProfileUsageMgr : itut_x780::ManagedObject {

     // Voir § 9.9.2.1 pour la description du comportement de cette
     // opération

     void reName (
          in NameType  oldProfileName,
          in NameType  newProfileName)
          raises (UnknownProfiles,
               AccessDenied,
               DuplicateProfileName
               );

     // Voir § 9.9.2.2 pour la description du comportement de cette
     // opération

     boolean inUse (
          in NameType  profileName)
          raises (UnknownProfiles,
          AccessDenied);

     // Voir § 9.9.2.3 pour la description du comportement de cette
     // opération

     void suspendUse (
          in NameType  profileName)
          raises (UnknownProfiles,
               AccessDenied);

     // Voir § 9.9.2.4 pour la description du comportement de cette
     // opération

     void resumeUse (
          in NameType   profileName)
          raises (UnknownProfiles, AccessDenied) ;

     // Voir § 9.9.2.5 pour la description du comportement de cette
     // opération

     void deleteProfile (
          in NameType profileName)
          raises (UnknownProfiles,
               AccessDenied,
               ProfileInUse);
```

```
    }; // interface ProfileUsageMgr

    /*
    Cet objet est instancié dans l'acteur appelé Répertoire d'objets de profil.
    Le système de gestion par fournisseur est le client.
    */


    interface ProfileRetriever : itut_x780::ManagedObject {

        // Voir § 9.9.3.1 pour la description du comportement de cette
        // opération

        ProfileInfoType retrieve (
            in NameType  profileName)
            raises (UnknownProfiles);

    }; // interface ProfileRetriever

}; // module ProfileManager

}; // module q834_4
#endif
```

C.10 Q834Registrar.idl

```
#ifndef __Q834_4_REGISTRAR_DEFINED
#define __Q834_4_REGISTRAR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module  q834_4
{

module Registrar
{

    // début des définitions issues d'autres fichiers en langage idl


    // de Q834Common
    typedef Q834Common::NameType          NameType;
    typedef Q834Common::ManagedEntityIdType    ManagedEntityIdType;
    typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
    typedef Q834Common::UserLabelType     UserLabelType;
    typedef Q834Common::SerialNumType     SerialNumType;
    typedef Q834Common::ReservationIdType      ReservationIdType;
    typedef Q834Common::DCNAddressType         DCNAddressType;
    typedef Q834Common::AdministrationDomainType     AdministrationDomainType;
    typedef Q834Common::UserLabelSeqType       UserLabelSeqType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define EquipmentFailure Q834Common::EquipmentFailure
#define InsufficientPONBW  Q834Common::InsufficientPONBW
#define InvalidSerialNumSyntax Q834Common::InvalidSerialNumSyntax
```

```
#define MaxSubtendingNodesExceeded Q834Common::MaxSubtendingNodesExceeded
#define UnknownNE Q834Common::UnknownNE
#define UnknownPort Q834Common::UnknownPort
#define DCNTimeout Q834Common::DCNTimeout
#define DeniedAccess Q834Common::DeniedAccess
#define BackupInProgress Q834Common::BackupInProgress
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define InvalidUserLabelSyntax Q834Common::InvalidUserLabelSyntax
#define SynchInProgress Q834Common::SynchInProgress

     // Fin des définitions issues d'autres fichiers en langage idl

     // Types de données locales

     // Exceptions locales
     exception TooManyNEs {};
     exception InvalidDCNAddress {};
     exception AddressLabelMismatch {};
     exception APONLayerFailure {};
     exception InvalidPort {};
     exception HWServicesMismatch {};

     // Fin des définitions locales

     valuetype NERegistrarValueType: itut_x780::ManagedObjectValueType {

          public ManagedEntityIdSeqType NEList; // GET
     };


     interface NERegistrar : itut_x780::ManagedObject {


          // Voir § 9.10.1.1 pour la description du comportement de cette
          // opération

          ManagedEntityIdType registerNE (
               in DCNAddressType nEDCNAddress,
               in UserLabelType nEUserLabel,
               in AdministrationDomainType administrationDomain)
               raises (AccessDenied,
                    DCNTimeout,
                    AddressLabelMismatch,
                    DuplicateUserLabel,
                    TooManyNEs,
                    InvalidDCNAddress,
                    CanNotAssignManagedEntityId,
                    CanNotRetrieveUserLabel,
                    DeniedAccess,
                    InvalidUserLabelSyntax);

          // Voir § 9.10.1.2 pour la description du comportement de cette
          // opération

          void modifyNEDCNAddress (
               in ManagedEntityIdType nEManagedEntityId,
               in DCNAddressType newNEDCNAddress)
               raises(AccessDenied,
                    DeniedAccess,
                    AddressLabelMismatch,
                    DCNTimeout,
                    CommFailure,
                    UnknownNE,
```

```
          InvalidDCNAddress,
          BackupInProgress);


// Voir § 9.10.1.3 pour la description du comportement de cette
// opération

ManagedEntityIdType  rangeONTorONU (
     in ManagedEntityIdType oltManagedEntityId,
     in UserLabelType nEUserLabel,
     in SerialNumType serialNum,
     in ManagedEntityIdType port ) // accès de terminaison OLT de
                                   //  réseau PON
     raises (AccessDenied,
          CommFailure,
          EquipmentFailure,
          UnknownNE,
          UnknownPort,
          MaxSubtendingNodesExceeded,
          InsufficientPONBW,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          DuplicateUserLabel,
          InvalidUserLabelSyntax,
          BackupInProgress,
          SynchInProgress );

// Voir § 9.10.1.4 pour la description du comportement de cette
// opération

ManagedEntityIdType rangeReplacementNE (
     in ManagedEntityIdType oldNEManagedEntityId,
     in UserLabelType newNEUserLabel,
     in SerialNumType replacementSerialNum )
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          EquipmentFailure,
          InvalidUserLabelSyntax,
          DuplicateUserLabel,
          HWServicesMismatch,
          BackupInProgress,
          SynchInProgress);

// Voir § 9.10.1.5 pour la description du comportement de cette
// opération

ManagedEntityIdType rangeUpgradeNE    (
     in ManagedEntityIdType oldNEManagedEntityId,
     in ManagedEntityIdType newNEManagedEntityId,
     in UserLabelType newNEUserLabel,
     in SerialNumType newNESerialNum )
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          EquipmentFailure,
          InvalidUserLabelSyntax,
          DuplicateUserLabel,
          HWServicesMismatch,
```

```
                    BackupInProgress,
                    SynchInProgress);

          // Voir § 9.10.1.6 pour la description du comportement de cette
          // opération

          ManagedEntityIdType moveONTorONU(
               in ManagedEntityIdType oldNEManagedEntityId,
               in ManagedEntityIdType newPONPort)
               raises (AccessDenied,
                    CommFailure,
                    UnknownPort,
                    APONLayerFailure,
                    EquipmentFailure,
                    InsufficientPONBW,
                    BackupInProgress,
                    SynchInProgress,
                    UnknownNE );

          // Voir § 9.10.1.7 pour la description du comportement de cette
          // opération

          ManagedEntityIdSeqType getSubtendingNEList(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, AccessDenied);

          // Voir § 9.10.1.8 pour la description du comportement de cette
          // opération

          ManagedEntityIdSeqType nEListGet ()
               raises (AccessDenied);


          // Voir § 9.10.1.9 pour la description du comportement de cette
          // opération

          void deRegisterNE(
               in ManagedEntityIdType nE)
               raises (UnknownNE,
                    AccessDenied);

          // Voir § 9.10.1.10 pour la description du comportement de cette
          // opération

          ManagedEntityIdType associateNE(
in ManagedEntityIdType preProvisionedNE,
               in ManagedEntityIdType discoveredNE)
               raises (UnknownManagedEntity,
                    AccessDenied);

     }; // interface NERegistrar

}; // module Registrar

}; // module q834_4

#endif
```

C.11 Q834ResourceAllocation.idl

```
#ifndef __Q834_4_RESOURCEALLOCATOR_DEFINED
#define __Q834_4_RESOURCEALLOCATOR_DEFINED
```

```
#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ResourceAllocation {

     // Début des définitions issues d'autres fichiers en langage idl

     // de Q834Common
     typedef Q834Common::ManagedEntityIdType          ManagedEntityIdType;
     typedef Q834Common::NameSeqType                  NameSeqType;
     typedef Q834Common::ReservationIdType            ReservationIdType;
     typedef Q834Common::ReservationIdSeqType   ReservationIdSeqType;
     typedef Q834Common::ServiceInstanceIdType  ServiceInstanceIdType;
     typedef Q834Common::EndPointType           EndPointType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define ConnectionCountExceeded Q834Common::ConnectionCountExceeded
#define InsufficientBW Q834Common::InsufficientBW
#define MaxSubtendingNodesExceeded Q834Common::MaxSubtendingNodesExceeded
#define UnknownNE Q834Common::UnknownNE
#define UnknownPort Q834Common::UnknownPort
#define UnknownProfiles Q834Common::UnknownProfiles
#define UnknownReservationId  Q834Common::UnknownReservationId
#define UnknownServiceInstance  Q834Common::UnknownServiceInstance
#define ProfileSuspended  Q834Common::ProfileSuspended

     // Fin de définitions issues d'autres fichiers en langage idl

     // Types de données locales


     struct BandwidthInfoType {
          long upstreamBW;
          long downstreamBW;
     };

     struct PortBandwidthType {
          BandwidthInfoType  portBandwidth;
          ManagedEntityIdType     portId;
     };

     typedef sequence<PortBandwidthType> PortBandwidthSeqType;

typedef PortBandwidthSeqType AvailableSysBandwidthSeqType;
     typedef PortBandwidthSeqType ReservedBandwidthSeqType;

     struct ReservationInfoType {
          ReservationIdType reservationId;
          EndPointType endPointA;
          EndPointType endPointB;
          NameSeqType profileNameList; // servicetemplates
          ServiceInstanceIdType serviceInstanceId;
          ReservedBandwidthSeqType reservedBandwidth;
     };

     struct ReservationBandwidthType {
          ReservationIdType reservationId;
          ReservedBandwidthSeqType reservedBandwidth;
     };
```

```
// Exceptions locales

// Fin des définitions locales


interface ResourceAllocator : itut_x780::ManagedObject {


     // Voir § 9.11.1.1 pour la description du comportement de cette
     // opération

     ReservationBandwidthType reserveForService(
          in EndPointType endPointA,
          in EndPointType endPointZ,
          in NameSeqType networkCharacteristicsProfiles,
          in ServiceInstanceIdType serviceInstanceId )
          raises (UnknownNE,
               UnknownPort,
               UnknownProfiles,
               InsufficientBW,
               MaxSubtendingNodesExceeded,
               ConnectionCountExceeded,
               CommFailure,
               AccessDenied,
               ProfileSuspended );

     // Voir § 9.11.1.2 pour la description du comportement de cette
     // opération

     AvailableSysBandwidthSeqType cancelReservation (
          in ReservationIdType reservationId )
          raises (UnknownReservationId,
               CommFailure,
               AccessDenied);

     // Voir § 9.11.1.3 pour la description du comportement de cette
     // opération

     ReservationIdType getReservationId (
          in ServiceInstanceIdType serviceInstanceId)
          raises (UnknownServiceInstance, AccessDenied);

     // Voir § 9.11.1.4 pour la description du comportement de cette
     // opération

     ReservedBandwidthSeqType reportReservedResources (
          in ManagedEntityIdType nEManagedEntityId)
          raises (UnknownNE, AccessDenied);

     // Voir § 9.11.1.5 pour la description du comportement de cette
     // opération

     ReservationIdSeqType getReservations(
          in ManagedEntityIdType nEManagedEntityId)
          raises (UnknownNE, AccessDenied);

     // Voir § 9.11.1.6 pour la description du comportement de cette
     // opération

     AvailableSysBandwidthSeqType cancelAllRemainingReservations(
          in ManagedEntityIdType nEManagedEntityId)
          raises (UnknownNE,
               CommFailure,
```

```
                    AccessDenied);

          // Voir § 9.11.1.7 pour la description du comportement de cette
          // opération

          ReservationInfoType getReservation (
               in ReservationIdType reservationId)
               raises (UnknownReservationId,
                    AccessDenied);

          // Voir § 9.11.1.8 pour la description du comportement de cette
          // opération


          AvailableSysBandwidthSeqType getAvaliableSysBandwidth(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE,
                    CommFailure,
                    AccessDenied);

     }; // interface ResourceAllocator


}; // module ResourceAllocation

}; // module q834_4

#endif
```

C.12 Q834SchedulerManagement.idl

```
#ifndef __Q834_4_SCHEDULERMGR_DEFINED
#define __Q834_4_SCHEDULERMGR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module SchedulerManagement {
     // Début des définitions issues d'autres fichiers en langage idl


     // de Q834Common

     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::OperationalStateType OperationalStateType;

#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidStartTime Q834Common::InvalidStartTime
#define InvalidStopTime Q834Common::InvalidStopTime

     // Fin des définitions issues d'autres fichiers en langage IDL

     // Types de données locales
```

```
enum HourlyDailyWeeklyMonthlyIndType {
     Hourly,
     Daily,
     Weekly,
     Monthly
};

enum DayofWeekType {
     Sunday,
     Monday,
     Tuesday,
     Wednesday,
     Thursday,
     Friday,
     Saturday,
     Unspecified
};

typedef short DayOfMonthType;  // 0 est interprété comme signifiant "non
                               //  spécifié".


struct TriggerTimeType
{
     long time; // Instant de déclenchement en nombre de secondes
     DayofWeekType dayOfWeek;
     DayOfMonthType dayofMonth;
};

typedef sequence<TriggerTimeType> TriggerTimeMatrixSeqType;

struct SchedulerType
{
     UserLabelType schedulerName;
     GeneralizedTimeType startTime;  // Instant de début du programme de
                                     //  planification
     GeneralizedTimeType stopTime;  // Instant d'arrêt du programme de
                                    //  planification
     HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd;
     TriggerTimeMatrixSeqType matrix;
     OperationalStateType OperationalState;
     AdministrativeStateType administrativeState;
};

typedef sequence<SchedulerType> SchedulerSeqType;

// Exceptions locales

exception MatrixSchedulerTypeMismatch {};
exception InvalidTrigger {};
exception ScheduleInUse {};

// Fin des définitions locales


valuetype SchedulerMgrValueType: itut_x780::ManagedObjectValueType {

     public SchedulerSeqType schedulerList; // GET

};

interface SchedulerMgr : itut_x780::ManagedObject {
```

```
        // Voir § 9.12.1.1 pour la description du comportement de cette
        // opération

        void makeScheduler (
             in UserLabelType schedulerName,
             in GeneralizedTimeType startTime,
             in GeneralizedTimeType stopTime,
             in HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd,
             in TriggerTimeMatrixSeqType matrix)
             raises (InvalidStartTime,
                  InvalidStopTime,
                  DuplicateUserLabel,
                  MatrixSchedulerTypeMismatch,
                  AccessDenied,
                  InvalidTrigger );


        // Voir § 9.12.1.2 pour la description du comportement de cette
        // opération

        void suspendScheduler (
             in UserLabelType schedulerName)
             raises (UnknownScheduler,
                  AccessDenied );

        // Voir § 9.12.1.3 pour la description du comportement de cette
        // opération

        void resumeScheduler (in UserLabelType schedulerName) raises
(UnknownScheduler, AccessDenied );


        // Voir § 9.12.1.4 pour la description du comportement de cette
        // opération

        void modifyTime (
             in UserLabelType schedulerName,
             in GeneralizedTimeType newStartTime,
             in GeneralizedTimeType newStopTime)
             raises (InvalidStartTime,
                  InvalidStopTime,
                  UnknownScheduler,
                  AccessDenied );


        // Voir § 9.12.1.5 pour la description du comportement de cette
        // opération

        void changeSchedulerName (
             in UserLabelType oldSchedulerName,
             in UserLabelType newSchedulerName)
             raises (UnknownScheduler,
                  DuplicateUserLabel,
                  AccessDenied );

        // Voir § 9.12.1.6 pour la description du comportement de cette
        // opération

        void modifyTriggerTimes (
             in UserLabelType schedulerName,
             in HourlyDailyWeeklyMonthlyIndType newHourlyDailyWeeklyMonthly,
             in TriggerTimeMatrixSeqType newMatrix)
             raises (UnknownScheduler,
                  MatrixSchedulerTypeMismatch,
```

```
                    AccessDenied,
                    InvalidTrigger );

          // Voir § 9.12.1.7 pour la description du comportement de cette
          // opération

          void removeScheduler (
               in UserLabelType schedulerName)
               raises (UnknownScheduler,
                    AccessDenied,
                    ScheduleInUse );

          // Voir § 9.12.1.8 pour la description du comportement de cette
          // opération

          SchedulerType retrieveScheduler (in UserLabelType schedulerName)
               raises (UnknownScheduler,
                    AccessDenied) ;

          // Voir § 9.12.1.9 pour la description du comportement de cette
          // opération

          SchedulerSeqType schedulerListGet () raises (AccessDenied);

     }; // interface SchedulerMgr

}; // module SchedulerManagement

}; // module q834_4

#endif
```

C.13 Q834ServiceProvisioning.idl

```
#ifndef __Q834_4_SERVICEPROVISIONING_DEFINED
#define __Q834_4_SERVICEPROVISIONING_DEFINED


#include "Q834Common.idl"
#include "Q834ProfileManager.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ServiceProvisioning {

// Début des définitions issues d'autres fichiers en langage idl

// de Q834Common
     typedef Q834Common::RDNType RDNType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::EndPointType EndPointType;
     typedef Q834Common::NameType NameType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define ConnectionCountExceeded Q834Common::ConnectionCountExceeded
#define EquipmentFailure Q834Common::EquipmentFailure
```

```
#define InsufficientBW Q834Common::InsufficientBW
#define InsufficientPONBW  Q834Common::InsufficientPONBW
#define UnknownProfiles Q834Common::UnknownProfiles
#define UnknownReservationId  Q834Common::UnknownReservationId
#define UnknownNE Q834Common::UnknownNE
#define UnknownServiceInstance Q834Common::UnknownServiceInstance
#define ProfileSuspended ProfileManager::ProfileSuspended
#define InvalidStartTime Q834Common::InvalidStartTime
#define InvalidStopTime Q834Common::InvalidStopTime
#define ParameterViolation Q834Common::ParameterViolation
#define UnknownPort Q834Common::UnknownPort
#define ProfileSuspended Q834Common::ProfileSuspended


// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales

// Exceptions locales
     exception UnknownConnection {};
     exception ConnectionAlreadyExists {};

// Fin des définitions locales


     interface ServiceProvisioner : itut_x780::ManagedObject {


          // Voir § 9.13.1.1 pour la description du comportement de cette
          // opération

          ManagedEntityIdType provisionConnection(
               in EndPointType endPointA,
               in EndPointType endPointB,
               in NameSeqType networkCharacteristicsProfiles,
               in ServiceInstanceIdType serviceInstanceId,
               in AdministrativeStateType administrativeState)
               raises (UnknownNE,
                    UnknownProfiles,
                    UnknownPort,
                    InsufficientBW,
                    ConnectionCountExceeded,
                    CommFailure,
                    EquipmentFailure,
                    ParameterViolation,
                    AccessDenied,
                    InsufficientPONBW,
                    ConnectionAlreadyExists,
                    ProfileSuspended);

          // Voir § 9.13.1.2 pour la description du comportement de cette
          // opération

          ManagedEntityIdType provisionReservation(
               in ReservationIdType reservationId,
               in AdministrativeStateType administrativeState)
               raises (UnknownReservationId,
                    AccessDenied);

          // Voir § 9.13.1.3 pour la description du comportement de cette
          // opération

          void deleteConnection (
               in ManagedEntityIdType subnetworkConnectionId)
```

```
                raises (UnknownConnection,
                     CommFailure,
                     EquipmentFailure,
                     AccessDenied);

          // Voir § 9.13.1.4 pour la description du comportement de cette
          // opération

          ManagedEntityIdType modifyConnection (
               in ManagedEntityIdType subnetworkConnectionId,
               in ManagedEntityIdType portB,
               in NameSeqType newNetworkCharacteristicsProfiles)
               raises (UnknownConnection,
                     UnknownPort,
                     UnknownProfiles,
                     InsufficientBW,
                     AccessDenied,
                     ProfileSuspended);

          // Voir § 9.13.1.5 pour la description du comportement de cette
          // opération

          void suspendService (
               in ServiceInstanceIdType serviceInstanceId,
               in GeneralizedTimeType startTime,
               in GeneralizedTimeType stopTime)
               raises (UnknownServiceInstance,
                     InvalidStartTime,
                     InvalidStopTime,
                     AccessDenied);

          // Voir § 9.13.1.6 pour la description du comportement de cette
          // opération

          void resumeService(
               in ServiceInstanceIdType serviceInstanceId)
               raises (UnknownServiceInstance,
                     AccessDenied);

     }; // interface ServiceProvisioner

}; // module ServiceProvisioning

}; // module q834_4

#endif
```

C.14 Q834Synchroniser.idl

```
#ifndef __Q834_4_SYNCHRONIZER_DEFINED
#define __Q834_4_SYNCHRONIZER_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module  q834_4
{

module Synchroniser
{

// début des définitions issues d'autres fichiers en langage idl
```

```
// de Q834Common
     typedef Q834Common::NameType     NameType;
     typedef Q834Common::UserLabelTypeUserLabelType;
     typedef Q834Common::ManagedEntityIdType    ManagedEntityIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define DCNTimeout Q834Common::DCNTimeout
#define EquipmentFailure Q834Common::EquipmentFailure
#define UnknownNE Q834Common::UnknownNE
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define BackupInProgress Q834Common::BackupInProgress
#define Timeout Q834Common::Timeout
#define SynchInProgress Q834Common::SynchInProgress


// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales

     typedef sequence<short> CurrentListingSeqType;

     struct ScheduledSynchNEType {
          ManagedEntityIdType managedEntityId;
          UserLabelType schedulerName;
     };
     typedef sequence<ScheduledSynchNEType> ScheduledSynchNESeqType;

// Exceptions locales
     exception NoSynchInProgress {};

// Fin de définitions locales

     valuetype SynchronizerValueType: itut_x780::ManagedObjectValueType {

          public ScheduledSynchNESeqType scheduledSynchNEList; // GET

     };


     interface NESynchroniser : itut_x780::ManagedObject {

          // définissent des constantes pour listes d'événements actuels

          const short CURRENT_ALARM             = 1;
          const short CURRENT_OPERATIONAL_STATE_DISABLED     = 2;
          const short CURRENT_ADMINSTRATIVE_STATE_LOCKED     = 3;
          const short CURRENT_PROTECTION_SWITCHING_EVENT     = 4;
          const short CURRENT_SERVICE_OUTAGE     = 5;

          // Voir § 9.14.1.1 pour la description du comportement de cette
          // opération

          void synchNE(
               in ManagedEntityIdType nEManagedEntityId)
               raises (AccessDenied,
                    CommFailure,
                    UnknownNE,
                    EquipmentFailure,
                    BackupInProgress,
```

```
          SynchInProgress);
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,

// Voir § 9.14.1.2 pour la description du comportement de cette
// opération

void abortSynchNE(
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          EquipmentFailure,
          NoSynchInProgress);

// Voir § 9.14.1.3 pour la description du comportement de cette
// opération

void scheduleSynchNE(
     in ManagedEntityIdType nEManagedEntityId,
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler);

// Voir § 9.14.1.4 pour la description du comportement de cette
// opération

void modifyNESynchSchedule(
     in ManagedEntityIdType nEManagedEntityId,
     in UserLabelType newSchedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler);

// Voir § 9.14.1.5 pour la description du comportement de cette
// opération

void cancelScheduledSynchNE(
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,
          UnknownNE);

// Voir § 9.14.1.6 pour la description du comportement de cette
// opération

void synchCurrentEventListings(
     in ManagedEntityIdType nEManagedEntityId,
     in CurrentListingSeqType currentListingTypeList)
     raises (AccessDenied,
          CommFailure,
          DCNTimeout,
          UnknownNE,
          EquipmentFailure,
          Timeout);

// Voir § 9.14.1.7 pour la description du comportement de cette
// opération

ScheduledSynchNESeqType scheduledSynchNEListGet ()
          raises (AccessDenied);
```

```
    }; // interface NESynchroniser

}; // module Synchroniser

}; // module q834_4

#endif
```

C.15 Q834Test.idl

```
#ifndef __Q834_4_TEST_DEFINED
#define __Q834_4_TEST_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module essai
{

    // début des définitions issues d'autres fichiers en langage idl

    // de Q834Common

    typedef Q834Common::ManagedEntityIdType    ManagedEntityIdType;
    typedef Q834Common::ManagedEntityIdSeqType  ManagedEntityIdSeqType;
    typedef Q834Common::UserIdType         UserIdType;
    typedef Q834Common::GeneralizedTimeType    GeneralizedTimeType;
    typedef Q834Common::UserLabelType      UserLabelType;
    typedef Q834Common::LoopbackLocationIdSeqType
    LoopbackLocationIdSeqType;
    typedef Q834Common::StatusValueType  StatusValueType;
    typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
    typedef Q834Common::TrackingObjectIdType TrackingObjectIdType;

#define AccessDenied             Q834Common::AccessDenied
#define CommFailure              Q834Common::CommFailure
#define UnknownScheduler         Q834Common::UnknownScheduler
#define InvalidScheduler         Q834Common::InvalidScheduler
#define InvalidStopTime          Q834Common::InvalidStopTime
#define InvalidStartTime         Q834Common::InvalidStartTime
#define UnknownManagedEntity     Q834Common::UnknownManagedEntity
#define UnknownNE                Q834Common::UnknownNE
#define UnknownServiceInstance   Q834Common::UnknownServiceInstance

    // Fin des définitions issues d'autres fichiers en langage idl


    // Types de données locales

    enum DirectionalityType {
            Egress,
        Ingress,
        BothDirections
    };
```

```
    struct ATMLoopbackInfoType {
         DirectionalityType directionality;
         LoopbackLocationIdSeqType   targetLLID;
         boolean segmentCellInd;
    };

    typedef  unsigned short TestIterationNumType;

    struct ATMLoopbackResultType  {
         LoopbackLocationIdSeqType  loopbackingLLID;
         unsigned long responseTime; //en µs
         boolean succeeded;          //Vrai ou Faux
    };

    typedef sequence<ATMLoopbackResultType> ATMLoopbackResultSeqType;

    struct  AggregateATMLoopbackResultType  {
         unsigned short  iterationSeqNum;
         ATMLoopbackResultSeqType  iterationTestResults;
    };

    typedef sequence<AggregateATMLoopbackResultType>
    AggregateATMLoopbackResultSeqType;

    struct ATMContinuityCheckInfoType {
         DirectionalityType directionality;
         boolean segmentCellInd;
    };


    typedef short DiagnosticType;    // Supplier specific.
    typedef DiagnosticType ResourceSelfTestInfoType;
    typedef sequence<ResourceSelfTestInfoType> ResourceSelfTestInfoSeqType;

    struct ResourceSelfTestResultType {
           DiagnosticType  diagnostic;
           boolean    testPassed;
    };

    typedef sequence<ResourceSelfTestResultType>
    ResourceSelfTestResultSeqType;
    typedef long long CCSetUpIdType;
    typedef TrackingObjectIdType TestTrackingObjectIdType;
    typedef long long LoopbackTrackingObjectIdType;


    typedef unsigned short LoopbackTestType;

    struct LoopbackInfoType {
         LoopbackTrackingObjectIdType trackingId;
         unsigned long  remainingTime; // in seconds
ManagedEntityIdType ctpId;
         LoopbackTestType loopbackTest;
         DirectionalityType directionality;
    };

    typedef sequence<LoopbackInfoType> LoopbackInfoSeqType;

    typedef string SupportedTestType;

    typedef sequence<SupportedTestType> SupportedTestSeqType;

    struct ScheduledTestNEType {
          ManagedEntityIdType managedEntityId;
```

```
       SupportedTestType     supportedTest;
       UserLabelType schedulerName;
};
typedef sequence<ScheduledTestNEType> ScheduledTestNESeqType;

struct TestHistoryType {
     ManagedEntityIdType managedEntityId;
     SupportedTestType  supportedTest;
     StatusValueType    testStatus;
};
typedef sequence<TestHistoryType> TestHistorySeqType;

// Exceptions locales

exception InvalidTimeoutPeriod {};
exception NotAvailableForTest {};
exception InvalidLocationId {};
exception UnknownTest {};
exception UncontrolledTestInProgress {};
exception InvalidDirection {};
exception InvalidTestOperations {};

// Fin des définitions locales

valuetype TestActionPerformerValueType: itut_x780::ManagedObjectValueType {

     public ScheduledTestNESeqType scheduledTestNEList; // GET

};


interface TestActionPerformer : itut_x780::ManagedObject {
     // Voir § 9.15.1.1 pour la description du comportement de cette
     // opération

     AggregateATMLoopbackResultSeqType aTMLoopback (
          in UserIdType testRequestorId,
          in ManagedEntityIdType ctp,
          in ATMLoopbackInfoType aTMLoopbackInfo,
          in TestIterationNumType testIterationNum,
          in ServiceInstanceIdType serviceInstanceId)
          raises (AccessDenied,
               CommFailure,
               UnknownManagedEntity,
               NotAvailableForTest,
               InvalidLocationId,
               InvalidDirection);

     // Voir § 9.15.1.2 pour la description du comportement de cette
     // opération

     CCSetUpIdType initializeContinuityCheck(
          in UserIdType testRequestorId,
          in ManagedEntityIdType sourceCtp,
          in ATMContinuityCheckInfoType aTMContinuityCheckInfo,
          in GeneralizedTimeType  stopTime,
          in ServiceInstanceIdType serviceInstanceId)
          raises (AccessDenied,
               CommFailure,
               UnknownManagedEntity,
               NotAvailableForTest,
               InvalidStartTime,
               InvalidStopTime,
               InvalidDirection);
```

```
// Voir § 9.15.1.3 pour la description du comportement de cette
// opération

void terminateContinuityCheck(
     in CCSetUpIdType cCSetUpId)
     raises (AccessDenied,
          CommFailure,
          UnknownTest);

// Voir § 9.15.1.4 pour la description du comportement de cette
// opération

TestTrackingObjectIdType scheduleResourceSelfTest(
     in UserIdType testRequestorId,
     in ManagedEntityIdType targetNE,
     in unsigned long timeOutPeriod, //In seconds.
     in ResourceSelfTestInfoSeqType specificTestInfo,
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler,
          InvalidTimeoutPeriod,
          InvalidTestOperations);


// Voir § 9.15.1.5 pour la description du comportement de cette
// opération

         void modifyResourceSelfTestSchedule(
           in TestTrackingObjectIdType testTrackingObjectId,
           in UserLabelType newSchedulerName)
           raises (AccessDenied,
                UnknownTest,
                UnknownScheduler,
                InvalidScheduler);

// Voir § 9.15.1.6 pour la description du comportement de cette
// opération

     void cancelScheduledResourceSelfTest (
          in TestTrackingObjectIdType testTrackingObjectId)
          raises (AccessDenied,
               UnknownTest);

// Voir § 9.15.1.7 pour la description du comportement de cette
// opération

TestTrackingObjectIdType conductResourceSelfTest (
     in UserIdType testRequestorId,
     in ManagedEntityIdType targetNE,
     in unsigned long timeOutPeriod, //In seconds.
     in ResourceSelfTestInfoSeqType specificTestInfo)
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          InvalidTimeoutPeriod,
          InvalidTestOperations);

// Voir § 9.15.1.8 pour la description du comportement de cette
// opération
```

```
ResourceSelfTestResultSeqType terminateResourceSelfTest (
     in TestTrackingObjectIdType testTrackingObjectId)
     raises (UnknownTest,
          UncontrolledTestInProgress,
          AccessDenied);

// Voir § 9.15.1.9 pour la description du comportement de cette
// opération

LoopbackTrackingObjectIdType initiateLoopback (
     in UserIdType testRequestorId,
     in ManagedEntityIdType loopingCtp,
     in long duration, //In minutes.
     in DirectionalityType directionality,
     in LoopbackTestType loopbackTest,
     in ServiceInstanceIdType serviceInstanceId)
     raises (AccessDenied,
          CommFailure,
          UnknownManagedEntity,
          NotAvailableForTest);

// Voir § 9.15.1.10 pour la description du comportement de cette
// opération

void terminateLoopback(
     in LoopbackTrackingObjectIdType  testTrackingObjectId)
     raises (UnknownTest,
          AccessDenied);

// Voir § 9.15.1.11 pour la description du comportement de cette
// opération


LoopbackInfoType getLoopbackInfo(
     in ManagedEntityIdType cTP)
     raises (UnknownManagedEntity,
          AccessDenied);

// Voir § 9.15.1.12 pour la description du comportement de cette
// opération

LoopbackInfoSeqType getLoopbackInfoByNE(
     in ManagedEntityIdType nEId)
     raises (UnknownManagedEntity,
          AccessDenied);

// Voir § 9.15.1.13 pour la description du comportement de cette
// opération

StatusValueType getTestStatus (
     in LoopbackTrackingObjectIdType id)
     raises (AccessDenied,
          UnknownTest);

// Voir § 9.15.1.14 pour la description du comportement de cette
// opération

ScheduledTestNESeqType  scheduledTestNEListGet()
     raises (AccessDenied);

// Voir § 9.15.1.15 pour la description du comportement de cette
// opération
```

```
          TestHistorySeqType testHistoryByManagedEntity (
               in ManagedEntityIdType managedEntityId)
               raises (UnknownManagedEntity,
                    AccessDenied);

          // Voir § 9.15.1.16 pour la description du comportement de cette
          // opération

          TestHistorySeqType testHistoryByServiceInstance (
               in ServiceInstanceIdType serviceInstanceId)
               raises (UnknownServiceInstance,
                    AccessDenied);


     }; // interface TestActionPerformer

}; // module Test

}; // module q834_4

#endif
```

C.16 Q834Filetransfer.idl

```
#ifndef __Q834_4_TRANSFERMGR_DEFINED
#define __Q834_4_TRANSFERMGR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module FileTransfer
{

// début des définitions issues d'autres fichiers en langage idl

// de Q834Common

     typedef Q834Common::ManagedEntityIdType    ManagedEntityIdType;
     typedef Q834Common::UserLabelType          UserLabelType;
     typedef Q834Common::StatusValueType   StatusValueType;
     typedef Q834Common::DCNAddressType         DCNAddressType;
     typedef Q834Common::FilenameType           FilenameType;
     typedef Q834Common::TransferTrackingObjectIdType
     TransferTrackingObjectIdType;
     typedef Q834Common::UserIdType                   UserIdType;
     typedef Q834Common::PasswordType           PasswordType;
     typedef Q834Common::GeneralizedTimeType    GeneralizedTimeType;

#define AccessDenied             Q834Common::AccessDenied
#define CommFailure              Q834Common::CommFailure
#define UnknownScheduler         Q834Common::UnknownScheduler
#define UnknownDestinationServer Q834Common::UnknownDestinationServer
#define InvalidScheduler         Q834Common::InvalidScheduler
#define UnknownRecordSet         Q834Common::UnknownRecordSet

// Fin des définitions issues d'autres fichiers en langage idl

// Types de données locales
```

```
    struct FileTransferHistoryType {
      ManagedEntityIdType recordSetId;
      DCNAddressType destinationServerAddr;
      UserIdType  userId;
      FilenameType destinationFile;
      GeneralizedTimeType transferTime;
    };
    typedef sequence<FileTransferHistoryType> FileTransferHistorySeqType;

    struct ScheduledFileTransferType {
      ManagedEntityIdType recordSetId;
      UserLabelType schedulerName;
    };
    typedef sequence<ScheduledFileTransferType> ScheduledFileTransferSeqType;

// Exceptions locales

    exception UnknownTransferProcess {};

// Fin des définitions locales

    valuetype TransferMgrValueType: itut_x780::ManagedObjectValueType {

        public FileTransferHistorySeqType fileTransferHistoryList; // GET
        public ScheduledFileTransferSeqType scheduledFileTransferList; // GET

    };

    interface TransferMgr : itut_x780::ManagedObject {

        // Voir § 9.16.1.1 pour la description du comportement de cette
        // opération

            TransferTrackingObjectIdType fileTransfer(
                in ManagedEntityIdType recordSetId,
                in DCNAddressType destinationServerAddr,
                in UserIdType userId,
                in PasswordType    password,
                in FilenameType destinationFile,
                in boolean overwriteExistingFile)
                raises (AccessDenied,
                    CommFailure,
                    UnknownRecordSet,
                    UnknownDestinationServer
                    );

        // Voir § 9.16.1.2 pour la description du comportement de cette
        // opération

            TransferTrackingObjectIdType scheduleFileTransfer(
                in ManagedEntityIdType recordSetId,
                in DCNAddressType destinationServerAddr,
                in UserIdType userId,
                in PasswordType    password,
                in FilenameType destinationFile,
                in boolean overwriteExistingFile,
                in UserLabelType schedulerName)
                raises (AccessDenied,
                    UnknownRecordSet,
                    UnknownDestinationServer,
                    UnknownScheduler,
                    InvalidScheduler);
```

```
        // Voir § 9.16.1.3 pour la description du comportement de cette
        // opération

                void modifyFileTransferSchedule(
                  in TransferTrackingObjectIdType transferTrackingObjectId,
                  in UserLabelType newSchedulerName)
                  raises (AccessDenied,
                       UnknownTransferProcess,
                       UnknownScheduler,
                       InvalidScheduler);

        // Voir § 9.16.1.4 pour la description du comportement de cette
        // opération

             void cancelScheduledFileTransfer (
                  in TransferTrackingObjectIdType transferTrackingObjectId)
                  raises (AccessDenied,
                       UnknownTransferProcess);

        // Voir § 9.16.1.5 pour la description du comportement de cette
        // opération

             StatusValueType getStatus(
                  in TransferTrackingObjectIdType transferTrackingObjectId)
                  raises (UnknownTransferProcess,
                       AccessDenied);

        // Voir § 9.16.1.6 pour la description du comportement de cette
        // opération

        FileTransferHistorySeqType fileTransferHistoryListGet ()
             raises (AccessDenied);

        // Voir § 9.16.1.7 pour la description du comportement de cette
        // opération

        ScheduledFileTransferSeqType  scheduledFileTransferListGet()
             raises (AccessDenied);

     }; // interface TransferMgr

}; // module FileTransfer

}; // module q834_4

#endif
```

Annexe D

Exemple de gabarits d'extrémité

Les Tableaux D.1 et D.2 ci-dessous donnent un exemple de gabarits montrant comment une extrémité peut être définie par un type de service. Le Tableau D.1 montre des exemples pour le cas où l'extrémité fait partie d'un accès d'interface NNI. Le Tableau D.2 montre des exemples pour le cas où l'extrémité fait partie d'un accès d'interface UNI.

Tableau D.1/Q.834.4 – Extrémités d'un accès d'interface NNI

Type de service	Accès	Paramètre	Profils
DS1	TDM DS3	Voie #	–
DS3	TDM DS3	–	–
DS1	ATM DS3	VPI/VCI	-
DS3	ATM DS3	VPI/VCI	-
DS1	TDM OC-n	STS #/VT1.5 #	-
DS1	ATM OC-n	VPI/VCI	-
DS3	ATM OC-n	VPI/VCI	-
LAN routé	ATM OC-n	VPI/VCI	-
LAN routé	ATM DS3	VPI/VCI	-
LAN routé	TDM DS3	Voie #	-
LAN routé	TDM OC-n	STS #/VT1.5 #	-
Voix	ATM DS3	VPI/VCI CID	TrafficDescriptorProfile (sens entrant) TrafficDescriptorProfile (sens sortant)
Voix	–	Groupe d'interface #/ Valeur de référence d'appel	-
ATM	ATM	VPI/VCI	TrafficDescriptorProfile (sens entrant) TrafficDescriptorProfile (sens sortant)

Tableau D.2/Q.834.4 – Extrémités d'un accès d'interface UNI

Type de service	Accès	Paramètre	Profils
DS1	TDM DS3	Voie #	DS1Profile CESProfile AAL1Profile
DS3	TDM DS3	-	DS3Profile CESProfile AAL1Profile
DS1	ATM DS3	VPI/VCI	–
DS3	ATM DS3	VPI/VCI	–
DS1	TDM OC-n or STS-n	STS #/VT1.5 #	AAL1Profile CESProfile
DS1	ATM OC-n	VPI/VCI	–
DS3	ATM OC-n	VPI/VCI	–
LAN routé	Ethernet	Accès logique #	BridgedLANServiceProfile AAL5Profile or AAL1Profile
Voix	RJ-11	-	VoiceServiceProfileAAL2 SSCSParameterProfile1 SSCSParameterProfile2 LESService
Voix	RJ-11	-	VoiceServiceProfileAAL1
ATM	ATM	VPI/VCI	TrafficDescriptorProfile (sens entrant) TrafficDescriptorProfile (sens sortant)
VLAN	Ethernet	Balise VLAN	–

SÉRIES DES RECOMMANDATIONS UIT-T

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

Imprimé en Suisse
Genève, 2004