UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T M.3120

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

Amendement 1
(05/2002)

SÉRIE M: RGT ET MAINTENANCE DES RÉSEAUX: SYSTÈMES DE TRANSMISSION, CIRCUITS TÉLÉPHONIQUES, TÉLÉGRAPHIE, TÉLÉCOPIE ET CIRCUITS LOUÉS INTERNATIONAUX

Réseau de gestion des télécommunications

Modèle générique informationnel d'architecture CORBA des réseaux et éléments de réseau

Amendement 1: commutation de protection

Recommandation UIT-T M.3120 (2001) – Amendement 1

RECOMMANDATIONS UIT-T DE LA SÉRIE M

RGT ET MAINTENANCE DES RÉSEAUX: SYSTÈMES DE TRANSMISSION, CIRCUITS TÉLÉPHONIQUES, TÉLÉGRAPHIE, TÉLÉCOPIE ET CIRCUITS LOUÉS INTERNATIONAUX

Introduction et principes généraux de maintenance et organisation de la maintenance	M.10–M.299
Systèmes de transmission internationaux	M.300–M.559
Circuits téléphoniques internationaux	M.560–M.759
Systèmes de signalisation à canal sémaphore	M.760–M.799
Systèmes internationaux de télégraphie et de phototélégraphie	M.800–M.899
Liaisons internationales louées par groupes primaires et secondaires	M.900–M.999
Circuits internationaux loués	M.1000–M.1099
Systèmes et services de télécommunication mobile	M.1100–M.1199
Réseau téléphonique public international	M.1200–M.1299
Systèmes internationaux de transmission de données	M.1300–M.1399
Appellations et échange d'informations	M.1400–M.1999
Réseau de transport international	M.2000–M.2999
Réseau de gestion des télécommunications	**M.3000–M.3599**
Réseaux numériques à intégration de services	M.3600–M.3999
Systèmes de signalisation par canal sémaphore	M.4000–M.4999

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

Recommandation UIT-T M.3120

Modèle générique informationnel d'architecture CORBA des réseaux et éléments de réseau

Amendement 1
Commutation de protection

Résumé

Le présent amendement propose des améliorations de la commutation de protection pour le modèle d'information de réseau générique. Le modèle décrit des classes d'objets gérés de commutation de protection ainsi que leurs propriétés qui sont génériques et utiles pour indiquer les informations échangées via toutes les interfaces définies pour l'architecture du RGT dans la Rec. UIT-T M.3010. Ces classes d'objets gérés génériques sont destinées à être appliquées indépendamment des technologies, des architectures et des services. On peut spécialiser les classes d'objets gérés de commutation de protection présentées dans le présent amendement pour appuyer la gestion de divers réseaux de télécommunication.

Source

L'Amendement 1 de la Recommandation M.3120 (2001) de l'UIT-T, élaboré par la Commission d'études 4 (2001-2004) de l'UIT-T, a été approuvé le 29 mai 2002 selon la procédure définie dans la Résolution 1 de l'AMNT.

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.

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

Recommandation UIT-T M.3120

Modèle générique informationnel d'architecture CORBA des réseaux et éléments de réseau

Amendement 1
Commutation de protection

1 Domaine d'application

Le présent amendement propose des améliorations du modèle de commutation de protection pour le modèle informationnel de réseau générique CORBA et de niveau NE. Le modèle IDL décrit des classes d'objets gérés de commutation de protection ainsi que leurs propriétés qui sont génériques et utiles pour décrire les informations échangées via toutes les interfaces définies pour l'architecture du RGT dans la Rec. UIT-T M.3010. Ces classes d'objets gérés génériques sont destinées à être appliquées indépendamment des technologies, des architectures et des services. Elles peuvent être étendues par divers secteurs industriels pour gérer des technologies réseau particulières, telles que l'ATM, la hiérarchie SDH SONET et l'Ethernet.

2 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.

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

[2] Recommandation UIT-T Q.816.1 (2001), *Services RGT à architecture CORBA: extensions pour la prise en charge des interfaces à granularité grossière*.

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

[4] Recommandation UIT-T X.780.1 (2001), *Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA à granularité grossière*.

[5] Recommandation UIT-T M.3120 (2001), *Modèle générique informationnel d'architecture CORBA des réseaux et éléments de réseau*.

[6] Recommandation UIT-T M.3100 (1995), *Modèle générique d'information de réseau, plus* Amendement 2 (2000).

[7] Recommandation ITU-T G.774 (2001), *Hiérarchie numérique synchrone – Modèle d'information de gestion du point de vue des éléments de réseau*.

[8] Recommandation UIT-T G.774.3 (2001), *Hiérarchie numérique synchrone – Gestion de la protection des sections multiplex du point de vue des éléments de réseau*.

3 Description générale du modèle informationnel de commutation de protection

Le paragraphe 4 de la présente Recommandation définit un ensemble d'interfaces CORBA pour le modèle d'information de commutation de protection. Ces interfaces sont traduites manuellement à partir d'un ensemble de classes d'objets gérés GDMO Rec. UIT-T M.3100 Amendement 2 et Rec. UIT-T G.774.3 en suivant le cadre CORBA du RGT et les directives données dans les Recs. UIT-T Q.816 et X.780 pour les interfaces CORBA à granularité fine.

Outre les interfaces à granularité fine décrites au § 4.5, un ensemble d'interfaces façade associées est défini au § 5.6. Ces interfaces façade sont définies conformément au cadre et aux directives donnés pour la granularité grossière dans les Recs. UIT-T Q.816.1 et X.780.1 pour la prise en charge des interfaces CORBA à granularité grossière. Le nom de ces interfaces de façade sont les noms des interfaces correspondant à une granularité fine suivis des caractères "_F" (soulignement suivi d'un F majuscule).

Le présent amendement fait partie intégrante de la Rec. UIT-T M.3120. Cela sous-entend que toutes les définitions (classes d'objets, type, structure, etc.) définies dans la Rec. UIT-T M.3120 sont dans le même module IDL et que l'on peut s'y référer sans l'identificateur de module.

Le module IDL du présent amendement a été rassemblé avec succès sans erreur de syntaxe. Le compilateur utilisé revendique la conformité CORBA 2.3 qui inclut le type de valeur et les capacités macro M4.

Les Figures 1 et 2 représentent les relations d'héritage, de confinement et d'association des interfaces CORBA définies dans la présente Recommandation. On notera que les interfaces façade suivent la même relation hiérarchique d'héritage que les interfaces à granularité fine correspondantes. Il faut également noter que des interfaces additionnelles, bien que n'étant pas requises, apparaissent dans les figures. Des exemples en sont les classes atelier.

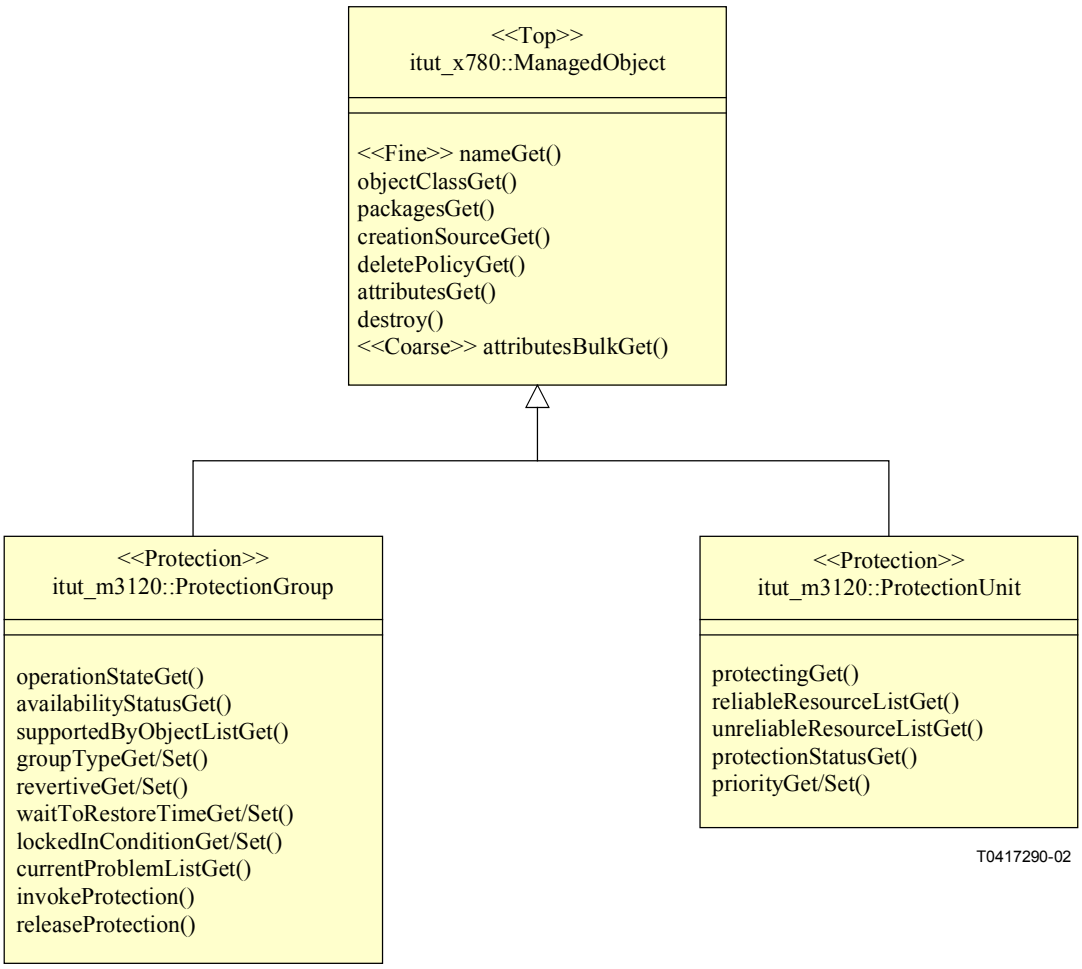

Figure 1/M.3120 – Relation d'héritage

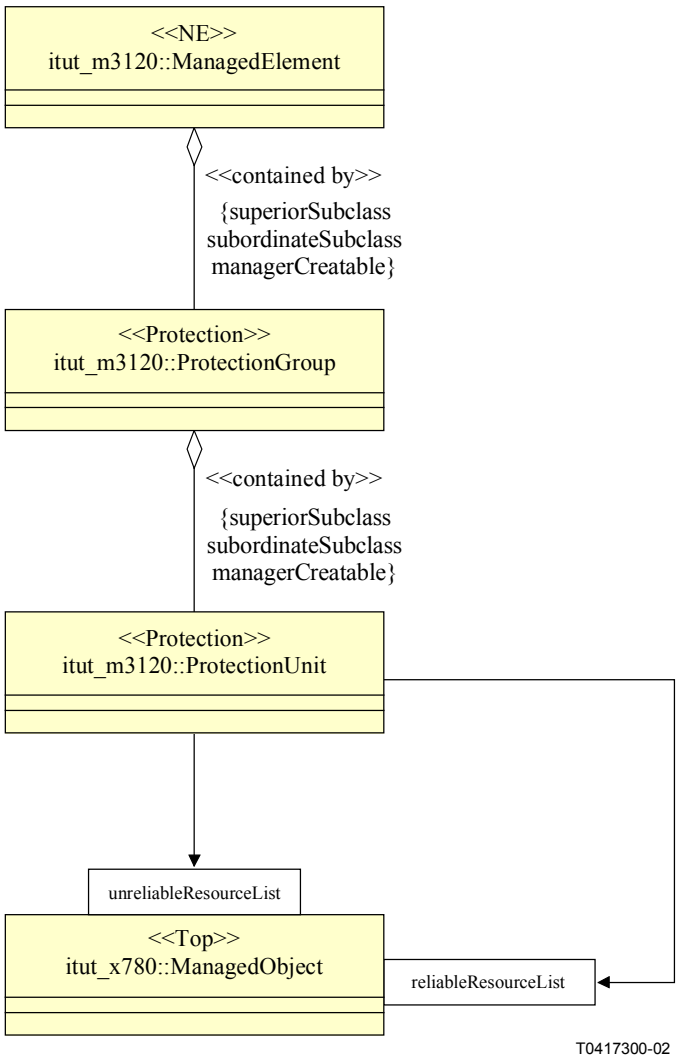

Figure 2/M.3120 – Relation de confinement et d'association

4 Représentation IDL du modèle d'information

```
#ifndef _itut_m3120_amd1_idl_
#define _itut_m3120_amd1_idl_


#include <itut_m3120.idl>


#pragma prefix "itu.int"


/**
Ce code IDL a pour but d'être enregistré dans un fichier nommé
"itut_m3120_amd1.idl" situé dans le trajet de recherche utilisé par les
compilateurs IDL sur votre système. Le module principal M.3120 (défini dans
M.3120) est contenu dans un fichier séparé appelé "itut_m3120.idl"
*/




/**
```

```
Ce fragment est ajouté au module, itut_m3120, qui contient une définition IDL
basée sur des objets définis dans M.3100 et G.855.1.
*/
module itut_m3120
{



/**
```

4.1 Importations

```
*/

/**
Types importés depuis itut_x780
*/
typedef itut_x780::AdditionalTextType AdditionalTextType;
typedef itut_x780::AdditionalInformationSetType AdditionalInformationSetType;
typedef itut_x780::CorrelatedNotificationSetType CorrelatedNotificationSetType;
typedef itut_x780::NotifIDType NotifIDType;
typedef itut_x780::NullType NullType;




/**
```

4.2 Déclarations aval

```
*/

/**
Déclarations aval d'interface
*/
     interface ProtectionGroup;
     interface ProtectionUnit;



/**
Déclarations aval Valuetype
*/
     valuetype ProtectionGroupValueType;
     valuetype ProtectionUnitValueType;


/**
Déclarations aval Typedefs
*/
     typedef MONameType ProtectionGroupNameType;
     typedef MONameType ProtectionUnitNameType;




/**
```

4.3 Structures et définitions de type

```
Les structures et les définitions de type définies ci-après sont classées par
ordre alphanumérique.
*/

     typedef sequence<ProtectionGroupNameType> ProtectionGroupNameSetType;

     typedef sequence<ProtectionGroupNameType> ProtectionGroupNameSeqType;

     typedef sequence<ProtectionUnitNameType> ProtectionUnitNameSetType;

     typedef sequence<ProtectionUnitNameType> ProtectionUnitNameSeqType;

/**
Cet attribut spécifie les critères de l'état immobilisé. Ces critères sont
notamment le taux de commutateur de protection automatique (APS, automatic
protection switching) ainsi qu'une fenêtre temporelle d'établissement et de
libération associée. Si le nombre de commutations APS d'une unité de protection
atteint la valeur spécifiée dans le champ hitsCount dans une lucarne temporelle
mobile de longueur spécifiée, l'unité de protection se mettra à l'état
immobilisé. Chaque commutation de protection et sa libération subséquente est
considérée comme une occurrence. La longueur de la fenêtre temporelle pour le
passage à l'état immobilisé est spécifiée dans le champ settingWindowTime. Dès
qu'une unité de protection est à l'état immobilisé, les demandes ultérieures de
commutation APS sont refusées jusqu'à ce que l'état immobilisé soit libéré. Le
critère de libération est l'absence de demande APS dans une autre fenêtre
temporelle. La longueur de celle-ci est spécifiée dans le champ
releasingWindowTime.
*/
     struct LockedInConditionType
     {
          unsigned short settingWindowTime;
          unsigned short releaseingWindowTime;
          unsigned short hitsCount;
     };

     enum ProtectionDirectionType
     {
          protectionDirectionTransmit,
          protectionDirectionReceive,
          protectionDirectionBidirectional
     };

     enum ProtectionGroupType
     {
          protectionGroupPlus, // 1+1 or hot-standby
          protectionGroupColon // M:N
     };

/**
L'attribut Protection Status (état de protection) donne l'état de la commutation
de protection dans un objet unité de protection. Le comportement qui s'applique
est le suivant:

–    Il doit avoir la capacité d'indiquer les demandes de commutation en suspens
     et actives relatives à l'unité de protection. Toutefois, une seule des
     valeurs interdiction, commutation forcée ou commutation manuelle peut être
     indiquée à la fois.

–    Un système de protection peut prendre en charge uniquement un sous-ensemble
     des valeurs acceptables pour cet attribut. Le sous-ensemble de valeurs que
     doit prendre en charge un système est spécifique à l'implémentation.
```

– La syntaxe de cet attribut comprend un sous-champ "relatedUnit" qui est une union de "fromProtectionUnitNumber" et "toProtectionUnitNumber". Ce sous-champ est utilisé pour indiquer l'unité sur laquelle le service est transporté.

Pour une PU protégée, les unités fromProtectionUnit et toProtectionUnit contiennent toutes deux l'identificateur de l'unité protégeante correspondante. Par la commutation sur la PU protégeante (c'est-à-dire que le service sera transporté par celle-ci), c'est le choix toProtectionUnit qui est utilisé; par la commutation de retour sur la PU protégée (le service sera transporté par celle-ci), c'est l'unité fromProtectionUnit.

Pour une PU protégeante, les numéros fromProtectionUnit et toProtectionUnit contiennent tous deux l'identificateur de l'unité protégée correspondante. Par la commutation sur la PU protégée (c'est-à-dire que le service sera transporté par celle-ci), c'est l'unité toProtectionUnit qui est utilisée; par la commutation de retour sur la PU protégeante (le service sera transporté par celle-ci), c'est l'unité fromProtectionUnit.

Si un système a la capacité de prendre en charge la commutation de protection pour dégradation de la ressource (RD, *resource degrade*) en plus de la commutation de protection pour défaillance de la ressource (RF, *resource fail*), la commutation de protection RD se déroule d'une manière analogue à celle décrite ci-après pour le cas de la défaillance RF.

Les valeurs admissibles suivantes de l'état de protection sont associées à chaque PU protégée:

- No Request (pas de demande): aucune demande de commutation n'est présente dans l'unité. Dans ce cas, le service est assuré par la PU protégée, la syntaxe d'état est pas de demande (noRequest). Dans le cas des systèmes non réversibles, la syntaxe d'état de l'unité protégeante correspondante est également pas de demande (noRequest).

- Manual Switch to Protecting Unit Complete (commutation manuelle sur unité protégeante exécutée): l'unité a effectué une commutation manuelle. Dans ce cas, le service est assuré par la PU protégeante correspondante, la syntaxe d'état de l'unité protégée est commande manuelle (manualSwitch) (switchStatus: completed; relatedUnit: toPU). La syntaxe d'état de l'unité protégeante correspondante est manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Release Failed (échec de déconnexion): une temporisation survient pendant l'attente de déconnexion. Dans ce cas, le service est toujours assuré par la PU protégeante, la syntaxe d'état est échec de déconnexion en plus de l'état précédent, par exemple commutation manuelle (manualSwitch) (switchStatus: completed; relatedUnit: toPU). La syntaxe d'état de l'unité protégeante correspondante est toujours l'état précédent, par exemple la commutation manuelle manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Automatic Switch (RF) Pending [commutation automatique (RF) en attente]: l'unité est à l'état de défaillance et l'unité protégeante n'est pas disponible. Dans ce cas, le service est assuré par la PU protégée, la syntaxe d'état est commutation automatique (autoSwitch) (switchStatus: pending; relatedUnit: to PU; reason: RF). La syntaxe d'état de l'unité protégeante correspondante est commutation automatique autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF) et son état précédent.

- Automatic Switch (RF) Complete [commutation automatique (RF) exécutée]: l'unité a commuté automatiquement sur l'unité protégeante comme conséquence d'un état de défaillance de l'équipement. Dans ce cas, le service est assuré par la PU protégeante correspondante, la syntaxe d'état de l'unité protégée est commutation automatique (autoSwitch) (switchStatus: completed; relatedUnit: toPU; reason: RF). La syntaxe d'état de l'unité protégeante correspondante est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF).

- Automatic Switch (RF) Present, Operate failed [commutation automatique (RF) présente, échec de fonctionnement]: une demande de commutation automatique (RF) est en cours et une temporisation est établie dans l'attente de l'exécution. Dans ce cas, le service est toujours assuré par la PU protégée, la syntaxe d'état est commutation automatique (autoSwitch) (switchStatus: failed; relatedUnit: toPU; reason: RF). La syntaxe d'état de l'unité protégeante correspondante est commutation automatique autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF) et son état précédent.

- Force Switch Complete, Automatic Switch (RF) Pending [commutation forcée exécutée, commutation automatique (RF) en attente]: l'unité a effectué une commutation forcée. Par ailleurs, l'unité a une commutation automatique (RF) en attente. Dans ce cas, le service est assuré par la PU protégeante correspondante, la syntaxe d'état de l'unité protégée est commutation forcée, forceSwitch (switchStatus: completed; relatedUnit: toPU) et la commutation automatique autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF). La syntaxe d'état de l'unité protégeante correspondante est commutation forcée, forceSwitch (switchStatus: completed; relatedUnit: fromPU) et la commutation automatique autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF).

- Automatic Switch Complete, Wait-To-Restore (revertive only) [commutation automatique exécutée, attente de rétablissement (fonctionnement réversible seulement)]: l'unité a commuté automatiquement sur l'unité protégeante. Dans ce cas, le service est assuré par l'unité protégeante correspondante, la syntaxe d'état de l'unité protégée est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: WTR). La syntaxe d'état de l'unité protégeante correspondante est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: WTR).

- Force Switch Complete (commutation forcée exécutée): l'unité a effectué une commutation forcée sur l'unité protégeante. Dans ce cas, le service est assuré par l'unité protégeante correspondante, la syntaxe d'état de l'unité protégée est commutation forcée (switchStatus: completed; relatedUnit: toPU). La syntaxe d'état de l'unité protégeante correspondante est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Protected Unit Lockout Completed (interdiction de l'unité protégée exécutée): l'unité a été exclue (interdite) de l'unité protégeante. Dans ce cas, le service est assuré par l'unité protégée et la syntaxe d'état est interdiction (switchStatus: completed).

- Protected Unit Lockout, Operate failed (exclusion de l'unité protégée, échec de fonctionnement): l'unité a été exclue de l'unité protégeante et la commutation effectuée précédemment n'a pas pu être libérée dans la limite de temporisation prévue. Lorsqu'une commutation est libérée, il est mis fin à l'échec de fonctionnement. Dans ce cas, le service est toujours assuré par l'unité protégeante correspondante, la syntaxe d'état de l'unité protégée est interdiction (switchStatus: completed) et échec de fonctionnement. La syntaxe d'état de l'unité protégeante correspondante est toujours l'état précédent, par exemple commutation manuelle manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Locked In (immobilisé): l'unité est dans l'état immobilisé. Celui-ci est causé par un excès d'événements de commutation de protection. Dans ce cas, le service est assuré par l'unité protégée, la syntaxe d'état est immobilisée.

Une unité protégée non réversible peut en outre avoir les valeurs d'état suivantes:

- Do Not Revert (ne pas inverser): l'unité protégée a été commutée sur l'unité protégeante et la demande de communication correspondante a été envoyée. La commutation sur l'unité protégeante est maintenue. Dans ce cas, le service est assuré par l'unité protégeante correspondante, la syntaxe d'état de l'unité protégée est ne pas inverser. La syntaxe d'état de l'unité protégeante correspondante est ne pas inverser.

- Manual Switch to Protected Unit Complete (commutation manuelle sur l'unité protégée exécutée): l'unité a effectué une commutation manuelle de l'unité protégeante sur l'unité protégée. Dans ce cas, le service est assuré par l'unité protégée, la syntaxe d'état est commutation manuelle manualSwitch (switchStatus: completed; relatedUnit: fromPU). La syntaxe d'état de l'unité protégeante correspondante est commutation manuelle (switchStatus: completed; relatedUnit: toPU).

- Force Switch to Protected Unit Complete (commutation forcée sur l'unité protégée exécutée): l'unité a effectué une commutation forcée de l'unité protégeante sur l'unité protégée. Dans ce cas, le service est assuré par l'unité protégée, la syntaxe d'état est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: fromPU). La syntaxe d'état de l'unité protégeante correspondante est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: toPU).

- Automatic Switch (RF) to Protected Unit Complete [commutation automatique (RF) sur l'unité protégée exécutée]: l'unité protégeante est à l'état de défaillance de l'équipement et le trafic est acheminé sur l'unité protégée. Dans ce cas, le service est assuré par l'unité protégée, la syntaxe d'état est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF). La syntaxe d'état de l'unité protégeante correspondante est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF).

- Force Switch from Protecting Unit Complete, Automatic Switch (RF) Pending [commutation forcée à partir de l'unité protégeante exécutée, commutation automatique (RF) en attente]: l'unité a effectué une commutation forcée de l'unité protégeante sur l'unité protégée. De plus, un état de commutation automatique (RF) est présent sur l'unité protégée. Dans ce cas, le service est assuré par l'unité protégée, la syntaxe d'état est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: fromPU) et commutation automatique autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF). La syntaxe d'état de l'unité protégeante correspondante est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: toPU) et commutation automatique autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF).

Les valeurs admissibles suivantes de l'attribut Protection Status sont associées à chaque unité de protection protégeante:

- No Request (pas de demande): aucune demande n'est présente dans l'unité. Dans ce cas, le service n'est pas assuré par l'unité protégeante et la syntaxe d'état est pas de demande noRequest. Dans un système non réversible, la syntaxe d'état de l'unité protégée correspondante est pas de demande noRequest.

- Manual Switch to Protecting Unit Complete (commutation manuelle sur l'unité protégeante exécutée): une unité protégée a effectué une commutation manuelle. Dans ce cas, le service est assuré par l'unité protégeante, la syntaxe d'état est commutation manuelle manualSwitch (switchStatus: completed; relatedUnit: fromPU). La syntaxe d'état de l'unité protégée correspondante est commutation manuelle manualSwitch (switchStatus: completed; relatedUnit: toPU).

- Automatic Switch (RF) Pending [commutation automatique (RF) en attente]: une unité protégée présente un état de défaillance de l'équipement et l'unité protégeante est indisponible pour la demande en question. Dans ce cas, le service est toujours assuré par l'unité protégée. La syntaxe d'état de l'unité protégeante correspondante est commutation automatique autoSwitch (switchStatus: pending; relatedUnit: fromPU; reason: RF) et son état précédent qui a causé l'indisponibilité. La syntaxe d'état de l'unité protégée correspondante est commutation automatique autoSwitch (switchStatus: pending; relatedUnit: toPU; reason: RF).

- Automatic Switch Complete (RF) to Protecting Unit [commutation automatique (RF) sur l'unité protégeante exécutée]: l'unité protégée a commuté automatiquement (RF) sur l'unité protégeante. Dans ce cas, le service est assuré par l'unité protégeante, la syntaxe d'état est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF). La syntaxe d'état de l'unité protégée correspondante est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF).

- Automatic Switch (RF) to Protecting Complete, Wait-To-Restore (revertive only) [commutation automatique (RF) exécutée, attente de rétablissement (en fonctionnement réversible seulement)]: l'unité a commuté automatiquement depuis l'unité protégée. Dans ce cas, le service est assuré par l'unité protégeante, la syntaxe d'état est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: WTR). La syntaxe d'état de l'unité protégée correspondante est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: WTR).

- Protecting Unit RF Present (état RF présent sur l'unité protégeante): l'unité protégeante est à l'état de défaillance de l'équipement. La syntaxe d'état de l'unité protégeante est ressource défaillante resourceFailed.

- Force Switch Complete to Protecting Unit (commutation forcée sur l'unité protégeante exécutée): l'unité a effectué une commutation forcée de l'unité protégée sur l'unité protégeante. Dans ce cas, le service est assuré par l'unité protégeante, la syntaxe d'état est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: fromPU). La syntaxe d'état de l'unité protégée correspondante est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: toPU).

- Protecting Unit Locked Out (interdiction de l'unité protégeante): l'unité protégeante a été interdite. Dans ce cas, le service n'est pas assuré par l'unité protégeante et la syntaxe d'état est interdiction (switchStatus: completed).

- Protecting Unit Release Lock Out Failed (défaillance de la libération de l'interdiction sur l'unité protégeante): la libération d'une interdiction est en cours mais n'a pas pu avoir lieu à la limite de la temporisation prévue. Dans ce cas, le service n'est pas assuré par l'unité protégeante et la syntaxe d'état est interdiction (switchStatus: failed).

Une unité de protection protégeante non réversible peut prendre les valeurs d'état additionnelles suivantes:

- Do Not Revert (ne pas inverser): l'unité protégée a été commutée sur l'unité protégeante et la demande de commutation correspondante a été envoyée. La commutation sur l'unité protégeante est maintenue. Dans ce cas, le service est assuré par l'unité protégeante, la syntaxe d'état est ne pas inverser doNotRevert. La syntaxe d'état de l'unité protégée correspondante est ne pas inverser doNotRevert.

- Manual Switch to Protected Unit Complete (commutation manuelle sur l'unité protégée exclue): l'unité a effectué une commutation manuelle de l'unité protégeante sur l'unité protégée. Dans ce cas, le service est assuré par l'unité protégée. La syntaxe d'état de l'unité protégeante est commutation manuelle manualSwitch (switchStatus: completed; relatedUnit: toPU). La syntaxe d'état de l'unité protégée correspondante est commutation manuelle manualSwitch (switchStatus: completed; relatedUnit: fromPU).

- Force Switch to Protected Unit Complete (commutation forcée sur l'unité protégée exécutée): l'unité protégeante a effectué une commutation forcée sur l'unité protégée. Dans ce cas, le service est assurée par l'unité protégée. La syntaxe d'état de l'unité protégeante est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: toPU). La syntaxe d'état de l'unité protégée correspondante est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Force Switch to Protected Unit Complete, Protecting Unit Equipment Failed (commutation forcée sur l'unité protégée exécutée, défaillance de l'équipement de l'unité protégeante): l'unité protégeante a effectué une commutation forcée sur l'unité protégée. De plus, l'unité protégeante est à l'état de défaillance de l'équipement. Dans ce cas, le service est assuré par l'unité protégée. La syntaxe d'état de l'unité protégeante est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: toPU) et défaillance d'équipement. La syntaxe d'état de l'unité protégée correspondante est commutation forcée forceSwitch (switchStatus: completed; relatedUnit: fromPU).

- Automatic Switch (RF) to Protected Unit Complete [commutation automatique (RF) sur l'unité protégée exécutée]: l'unité protégeante est à l'état de défaillance de l'équipement et l'unité protégée est maintenant utilisée. Dans ce cas, le service est assuré par l'unité protégée. La syntaxe d'état de l'unité protégeante est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: toPU; reason: RF). La syntaxe d'état de l'unité protégée correspondante est commutation automatique autoSwitch (switchStatus: completed; relatedUnit: fromPU; reason: RF).

*/

```
enum PAutoSwitchReasonType
{
     pAutoSwitchReasonWaitToRestore,
     pAutoSwitchReasonSignalDegrade,
     pAutoSwitchReasonSignalFail
};

enum PRelatedUnitChoiceType
{
     pFrom,
     pTo
};

enum PSwitchStatusType
{
     pSwitchStatusPending,
     pSwitchStatusCompleted,
```

```
     pSwitchStatusFailed
};

enum PLockoutChoiceType
{
     pSwitchStatus,
     pReleaseFailed
};

union PRelatedUnitType switch (PRelatedUnitChoiceType)
{
     case pFrom:
        ProtectionUnitNameType fromUnit;
     case pTo:
        ProtectionUnitNameType toUnit;
};

struct PManualOrForcedSwitchType
{
     PSwitchStatusType      switchStatus;
     PRelatedUnitType       relatedUnit;
};

struct PAutoSwitchType
{
     PSwitchStatusType      switchStatus;
     PRelatedUnitType       relatedUnit;
     PAutoSwitchReasonType  autoSwitchReason;
};

union PLockoutType switch (PLockoutChoiceType)
{
     case pSwitchStatus:
        PSwitchStatusType   switchStatus;
     case pReleaseFailed:
        NullType            releaseFailed;

};

enum ProtectionStatusChoiceType
{
     protectionStatusNoRequest,
     protectionStatusDoNotRevert,
     protectionStatusManualSwitch,
     protectionStatusAutoSwitch,
     protectionStatusForcedSwitch,
     protectionStatusLockout,
     protectionStatusReleaseFailed,
     protectionStatusResourceFailed,
     protectionStatusLockedIn
};

union ProtectionStatusType switch (ProtectionStatusChoiceType)
{
     case protectionStatusNoRequest:
        NullType       noRequest;
     case protectionStatusDoNotRevert:
        NullType       doNotRevert;
     case protectionStatusManualSwitch:
        PManualOrForcedSwitchType    maunalSwitch;
     case protectionStatusAutoSwitch:
        PAutoSwitchType              autoSwitch;
     case protectionStatusForcedSwitch:
        PManualOrForcedSwitchType    forcedSwitch;
```

```
        case protectionStatusLockout:
           PLockoutType                 lockout;
        case protectionStatusReleaseFailed:
           NullType       releaseFailed;
        case protectionStatusResourceFailed:
           NullType       resourceFailed;
        case protectionStatusLockedIn:
           NullType       lockedIn;
    };

    struct ProtectionStatusType
    {
        ProtectionStatusChoiceType      choice;
        PSwitchStatusType               switchStatus;
           // may be NULL
        PRelatedUnitType                relatedUnit;
           // may be NULL
        PAutoSwitchReasonType           autoSwitchReason;
           // may be NULL
    };

    typedef sequence<ProtectionStatusType> ProtectionStatusSetType;

    enum PSwitchActionResultType
    {
        pSwitchActionResultPreempted,
        pSwitchActionResultFailure,
        pSwitchActionResultTimeout
    };

    enum PSwitchType
    {
        pSwitchManual,
        pSwitchForced,
        pSwitchLocked
    };


/**
```

4.4 Exceptions

```
*/

/**
```

4.4.1 Exceptions et constantes pour paquetages conditionnels

```
*/

    exception NOpriorityPackage {};

    exception NOprotectionAlarmPackage {};

/**
Cet ensemble de constantes représente les paquetages conditionnels définis dans
la présente Recommandation. Lorsqu'un paquetage conditionnel est pris en charge
dans une instance d'objet géré donnée, une des chaînes sera contenue dans
l'attribut de paquetage.
*/
      const string priorityPackage =
          "itut_m3120::priorityPackage";
      const string protectionAlarmPackage =
         "itut_m3120::protectionAlarmPackage";

/**
```

4.5 Interfaces – A granularité fine

*/

/**

4.5.1 ProtectionGroup

Cette classe d'objets sert à représenter un système de protection au sein d'un élément de réseau (NE, *network element*). La notification d'événements de commutation de protection et le contrôle du système de gestion relatifs aux interdictions, commutations forcées et commutations manuelles sont les principales fonctions de gestion assurées par l'entité en question.

Des instances de cette classe d'objets peuvent être automatiquement créées dans un agent, par exemple immédiatement après l'initialisation des ressources d'élément NE intervenant dans le système de protection, compte tenu de la composition et du mode de l'élément NE. Des instances de cette classe d'objets peuvent être automatiquement libérées dans l'agent.

Un élément de réseau NE peut comporter plusieurs instances de l'objet protectionGroupR2 (une par système de protection assuré par l'élément NE). Une instance de l'objet ProtectionGroup contiendrait deux ou plusieurs instances de l'objet ProtectionUnit.

Toutes les instances de l'unité de protection à l'intérieur d'un objet groupe de protection auront le paquetage priorityPkg, ou aucune ne doit l'avoir. Il faut noter qu'avant la création de l'objet ProtectionGroup, l'attribut liste d'objets pris en charge (sbol) d'une ressource fiable telle qu'un objet point de terminaison peut indiquer un objet ressource non fiable tel qu'une carte de circuit imprimé. Mais dès qu'un objet groupe de protection est créé, l'attribut sbol désigne l'objet groupe de protection.

Cette classe d'objets a les attributs suivants:

– Operational State (état opérationnel): attribut accessible en lecture qui indique si le mécanisme de protection représenté par cette instance a la capacité de remplir ses fonctions normales.

– Protection Group Type (type de groupe de protection): attribut accessible en lecture qui indique si la méthode de protection utilisée est à structure doublée (1+1) ou M:N.

– Revertive (réversible): attribut accessible en lecture qui indique si le système de protection utilisé est réversible. Sa valeur par défaut indiquera le fonctionnement réversible, mais il doit pouvoir être mis à la valeur correspondant au fonctionnement non réversible par une commande du gestionnaire.

– Wait-to-Restore Time (temps d'attente de rétablissement): attribut accessible en lecture et en écriture qui indique le temps, en secondes, que le système de protection doit attendre après la suppression d'un dérangement pour se commuter à nouveau sur la ressource protégée. Cet attribut ne présente de l'intérêt qu'en fonctionnement réversible.

– LockedInCondition: cet attribut accessible en lecture et en écriture spécifie les critères de l'état immobilisé.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants:

– protectionAlarmPackage: présent si le système a la capacité de signaler la défaillance du mécanisme de protection ou de la ressource de protection.

```
–    createDeleteNotificationsPackage: présent si une instance le prend en
     charge.

–    attributeValueChangeNotificationPackage: présent si une instance le prend
     en charge.

*/
     valuetype ProtectionGroupValueType: itut_x780::ManagedObjectValueType
     {
          public OperationalStateType operationalState;
             // GET
          public AvailabilityStatusSetType availabilityStatus;
             // GET
          public MONameSetType             supportedByObjectList;
             // GET-REPLACE, ADD-REMOVE
          public ProtectionGroupType       groupType;
             // GET-REPLACE
          public boolean                   revertive;
             // GET-REPLACE-WITH-DEFAULT
          public unsigned short            waitToRestoreTime;
             // GET-REPLACE
          public LockedInConditionType     lockedInCondition;
             // GET-REPLACE
          public CurrentProblemSetType     currentProblemList;
             // conditional
             // protectionAlarmPackage
             // GET
     }; // valuetype ProtectionGroupValueType


     interface ProtectionGroup: itut_x780::ManagedObject
     {
          const boolean revertiveDefault = TRUE;

          OperationalStateType operationalStateGet ()
             raises (itut_x780::ApplicationError);

          AvailabilityStatusSetType availabilityStatusGet ()
             raises (itut_x780::ApplicationError);


          MONameSetType supportedByObjectListGet ()
             raises (itut_x780::ApplicationError);

          void supportedByObjectListSet
             (in MONameSetType objectList)
             raises (itut_x780::ApplicationError);

          void supportedByObjectListAdd
             (in MONameSetType objectList)
             raises (itut_x780::ApplicationError);

          void supportedByObjectListRemove
             (in MONameSetType objectList)
             raises (itut_x780::ApplicationError);


          ProtectionGroupType groupTypeGet ()
             raises (itut_x780::ApplicationError);


          boolean revertiveGet ()
             raises (itut_x780::ApplicationError);
```

```
        void revertiveSet
            (in boolean revertive)
            raises (itut_x780::ApplicationError);


        unsigned short waitToRestoreTimeGet ()
            raises (itut_x780::ApplicationError);

        void waitToRestoreTimeSet
            (in unsigned short waitTime)
            raises (itut_x780::ApplicationError);


        LockedInConditionType lockedInConditionGet ()
            raises (itut_x780::ApplicationError);

        void lockedInConditionSet
            (in LockedInConditionType condition)
            raises (itut_x780::ApplicationError);


        CurrentProblemSetType currentProblemListGet ()
            raises (itut_x780::ApplicationError,
            NOprotectionAlarmPackage);

/**
Invoke Protection (appel de protection): action servant à demander une
interdiction, une commutation forcée ou une commutation manuelle au niveau de
l'une ou de plusieurs instances ProtectionUnit contenues dans l'objet
ProtectionGroup. Les paramètres d'entrée suivants sont inclus dans l'action
appel de protection:

–    type de commutation (manuelle, forcée ou interdite);

–    entité de protection (facultative): identificateur(s) de l'unité protégée
     et/ou protégeante à laquelle s'applique l'appel.

En l'absence de celle-ci, l'appel est censé s'appliquer à toutes les entités de
cette nature dans le groupe de protection.

Si une unité de protection est identifiée dans le champ protectedUnits ou si une
unité protégée est identifiée dans le champ protectingUnits field, l'action
échoue.

Si la demande est une commutation forcée ou une commutation manuelle le champ
protectedUnits identifiera une ou plusieurs unités de protection. Si une seule
unité est identifiée dans le champ protectedUnits field et qu'il n'y a qu'une
seule unité de protection dans le groupe de protection, le champ protectingUnits
peut être omis. Si le champ protectingUnits est présent, il identifie le même
nombre d'unités que le champ protectedUnits.

Si la demande est une commutation interdite, le champ entité de protection peut
être absent, indiquant ainsi que la demande s'applique à toutes les entités de
protection contenues. Si le champ entité de protection est présent, n'importe
quel nombre d'unités de protection peut être identifié dans le champ
protectedUnits et/ou protectingUnits, et chacun de ceux-ci peut être absent.

Dans le cas d'une demande de commutation interdite, les unités protégeantes
et/ou les unités protégées sont interdites.
```

```
Dans le cas des demandes qui ne peuvent être satisfaites, soit parce que la
demande s'applique à une unité protégeante répondant à une demande ayant une
priorité plus élevée, soit en raison d'une défaillance (failure), soit en raison
d'une temporisation (timeout), la réponse indiquera la raison pour laquelle la
demande n'a pas été satisfaite et la demande ne sera pas rendue pendante.

*/
       void invokeProtection
              (in PSwitchType switchType,
              in ProtectionUnitNameSeqType protectedUnitList,
              in ProtectionUnitNameSeqType protectingUnitList,
              out PSwitchActionResultType actionResult)
              raises (itut_x780::ApplicationError);

/**
Release Protection (déconnexion de protection): action servant à arrêter une
interdiction, une commutation forcée ou une commutation manuelle sur une ou
plusieurs des instances protectionUnit contenues dans l'objet protectionGroup.
Les paramètres d'entrée suivants sont inclus dans l'action Release Protection:

–    type de commutation (manuelle, forcée ou interdite);

–    entité de protection (facultative): identificateur(s) de l'unité protégée
     et/ou protégeante à laquelle s'applique l'appel.

En l'absence de cette unité, l'appel est censé s'appliquer à toutes les entités
de cette nature dans le groupe de protection.

Si une unité de protection est identifiée dans le champ protectedUnits ou si une
unité protégée est identifiée dans le champ protectingUnits, l'action échoue.

Si la demande est une commutation forcée ou une commutation manuelle, le champ
protectedUnits identifiera une ou plusieurs unités de protection, et le champ
protectingUnits sera omis. Pour chaque unité protégée identifiée, à moins
qu'elle ne soit commutée sur une unité protégeante, l'action échoue.

Si la demande est une commutation interdite, le champ protectionEntity peut être
absent, indiquant ainsi que la demande s'applique à toutes les entités de
protection contenues. Si le champ protectionEntity est présent, n'importe quel
nombre d'unités de protection peut être identifié dans le champ protectedUnits
et/ou protectingUnits, et chacun de ceux-ci peut être absent.

Dans le cas d'une demande de commutation interdite, les unités protégeantes
et/ou les unités protégées cessent d'être interdites. Autrement dit, les unités
protégées sont maintenant sous protection et les unités protégeantes sont
maintenant capables d'assurer une protection.

Dans le cas des demandes de libération qui ne peuvent être satisfaites, la
réponse indiquera la raison pour laquelle la demande n'a pas été satisfaite.

*/
       void releaseProtection
          (in PSwitchType switchType,
          in ProtectionUnitNameSeqType protectedUnitList,
          in ProtectionUnitNameSeqType protectingUnitList,
          out PSwitchActionResultType actionResult)
          raises (itut_x780::ApplicationError);


       MANDATORY_NOTIFICATION(
              Notifications, protectionSwitchReporting)
       MANDATORY_NOTIFICATION(
              itut_x780::Notifications, stateChange)
```

```
  CONDITIONAL_NOTIFICATION(
        Notifications, protectionAlarm,
        protectionAlarmPackage)
  CONDITIONAL_NOTIFICATION(
        itut_x780::Notifications, objectCreation,
        createDeleteNotificationsPackage)
  CONDITIONAL_NOTIFICATION(
        itut_x780::Notifications, objectDeletion,
        createDeleteNotificationsPackage)
  CONDITIONAL_NOTIFICATION(
        itut_x780::Notifications, attributeValueChange,
        attributeValueChangeNotificationPackage)

};  // interface ProtectionGroup


interface ProtectionGroupFactory: itut_x780::ManagedObjectFactory
{
  itut_x780::ManagedObject create
     (in NameBindingType nameBinding,
     in MONameType superior,
     in string reqID,   // auto naming if empty string
     out MONameType name,
     in StringSetType packageNameList,
     in MONameSetType supportedByObjectList,
      // GET-REPLACE, ADD-REMOVE
     in ProtectionGroupType groupType,
      // GET-REPLACE
     in boolean revertive,
      // GET-REPLACE-WITH-DEFAULT
     in unsigned short waitToRestoreTime,
      // GET-REPLACE
     in LockedInConditionType lockedInCondition)
      // GET-REPLACE
     raises (itut_x780::ApplicationError,
      itut_x780::CreateError);

}; // interface ProtectionGroupFactory
```

/**

4.5.2 ProtectionUnit

La classe d'objets gérés ProtectionUnit sert à représenter la ressource protégée (en service, ordinaire ou préférée) ou protégeante (secours ou réserve) dans un système de protection. Il met en relation des ressources (telles que des cartes de circuits) concernées par le système de protection et suit l'état de la commutation de protection des ressources.

Des instances de cette classe d'objets peuvent être instanciées par l'agent en fonction des systèmes de protection adoptés par l'élément NE (tel qu'une carte de circuits). Une instance unité de protection est supprimée à la suppression de l'instance objet ressource désignée par l'attribut liste de ressouces non fiables et peut être créée automatiquement à la création de l'objet ressource associé. L'agent peut aussi créer et supprimer des instances de cette classe d'objets pour refléter les modifications locales du système de protection. La notification attributeValueChange sert à notifier les changements des attributs liste de ressources fiables, état de protection et priorité.

Deux ou plusieurs instances de l'objet ProtectionUnit peuvent exister dans une instance de l'objet ProtectionGroup.

Une instance de l'objet ProtectionUnit peut contenir une instance de l'objet ProtectionCurrentData.

Une instance de ProtectionUnit est liée à des instances d'entités de ressource (cartes de circuit, par exemple) via l'attribut liste de ressources non fiable. Si la fonction des entités de ressource (fonction de rythme, fonction point de terminaison du transport, etc.) est explicitement modélisée en tant qu'instance d'objet dans l'élément NE, une instance ProtectionUnit est également liée aux instances de l'entité de fonction modélisée via l'attribut liste de ressources fiables.

Cette classe d'objets a les attributs suivants:

– Protecting (protégeant): attribut accessible en lecture qui indique si l'unité de protection est associée ou non à une ressource assurant un rôle protecteur ("Vrai") ou un rôle protégé ("Faux") dans le système de protection.

– Unreliable Resource List (liste de ressources non fiables): attribut accessible en lecture qui désigne la ressource non fiable (entité carte de circuit par exemple) associée à l'objet unité de protection (par exemple la ressource effectivement protégée ou protégeante). La syntaxe de cet attribut est évaluée sur un ensemble et peut désigner des instances multiples de ressources non fiables lorsqu'un ensemble de ressources forme une unité atomique dans le système de protection.

– Reliable Resource List (liste de ressources fiables): attribut accessible en lecture qui désigne la ressource fiable (c'est-à-dire l'entité fonctionnelle), s'il y en a une, associée à l'unité de protection. La valeur de cet attribut change lorsque l'unité de protection (PU, *protection unit*) intervient dans une commutation ou une déconnexion de protection. Cet attribut désigne pour une unité PU protégée, lorsque celle-ci n'est pas commutée, la ressource fiable associée (c'est-à-dire l'objet fonctionnel); lorsqu'elle est commutée, cet attribut désigne NULL. Dans le cas d'une unité PU protégeante, cet attribut désigne NULL lorsqu'elle n'est pas commutée et désigne la ressource fiable associée (c'est-à-dire l'objet fonctionnel) lorsqu'elle est commutée. La syntaxe de cet attribut est évaluée sur un ensemble et peut désigner plusieurs instances de ressources fiables lorsqu'un ensemble d'objets fonctionnels forme une unité atomique dans le système de protection.

– Priority: attribut accessible en lecture et en écriture qui spécifie la priorité du service transporté sur la ressource associée à l'instance PU. Des valeurs acceptables de cet attribut sont des entiers, la priorité la plus élevée étant indiquée par la valeur 1, une priorité plus faible par une valeur plus grande.

– Protection Status R1 (état de protection R1): attribut accessible en lecture qui donne l'état de la commutation de protection dans un objet unité de protection. Le comportement qui s'applique est le suivant:

  • Il doit avoir la capacité d'indiquer les demandes de commutation en suspens et actives relatives à l'unité de protection. Toutefois, une seule des valeurs interdiction, commutation forcée ou commutation manuelle peut être indiquée à la fois;

  • Un système de protection peut prendre en charge uniquement un sous-ensemble des valeurs acceptables pour cet attribut. Le sous-ensemble de valeurs que doit prendre en charge un système est spécifique à l'implémentation.

```
Cette interface contient les PAQUETAGES CONDITIONNELS ci-après.

–    priorityPackage: présent si une instance le prend en charge.

–    attributeValueChangeNotificationPackage: présent si une instance le prend
     en charge.
*/
     valuetype ProtectionUnitValueType: itut_x780::ManagedObjectValueType
     {
          public boolean                  protecting;
              // GET
          public MONameSetType            reliableResourceList;
              // GET
          public MONameSetType            unreliableResourceList;
              // GET
          public ProtectionStatusSetType  protectionStatus;
              // GET
          public unsigned short           priority;
              // conditional
              // priorityPackage
              // GET-REPLACE
     }; // valuetype ProtectionUnitValueType


     interface ProtectionUnit : itut_x780::ManagedObject
     {
       boolean protectingGet ()
          raises (itut_x780::ApplicationError);

/**

La valeur de l'attribut reliableResourceList attribute désigne la ou les
ressources fiables (les objets fonctionnels) qui sont associées à l'instance
unité de protection.
*/
       MONameSetType reliableResourceListGet ()
          raises (itut_x780::ApplicationError);

/**
La valeur de l'attribut unreliableResourceList désigne la ou les ressources
fiables (carte de circuits, p. ex.) qui sont associées à l'instance unité de
protection.
*/
       MONameSetType unreliableResourceListGet ()
          raises (itut_x780::ApplicationError);

/**
*/
       ProtectionStatusSetType protectionStatusGet ()
          raises (itut_x780::ApplicationError);

/**
Cet attribut spécifie la priorité du service acheminé sur la ressource associée
à l'instance Protection Unit. Des valeurs valables de cet attribut sont des
entiers, la priorité la plus élevée étant indiquée par la valeur 1, une priorité
plus faible par une valeur plus grande.

Pour une unité ProtectionUnit protégeante, la valeur de cet attribut indique la
priorité du choix de cette unité par rapport à une ou plusieurs autres unités
protégeantes dans le même ProtectionGroup. Plus la valeur est petite, plus
l'unité ProtectionUnit est préférée par rapport à d'autres ProtectionUnit.
```

```
*/
      unsigned short priorityGet ()
         raises (itut_x780::ApplicationError,
              NOpriorityPackage);

      void prioritySet
         (in unsigned short priority)
         raises (itut_x780::ApplicationError,
              NOpriorityPackage);

      CONDITIONAL_NOTIFICATION(
         itut_x780::Notifications, attributeValueChange,
         attributeValueChangeNotificationPackage)

    }; // interface ProtectionUnit


    interface ProtectionUnitFactory: itut_x780::ManagedObjectFactory
    {
      itut_x780::ManagedObject create
         (in NameBindingType nameBinding,
         in MONameType superior,
         in string reqID,   // auto naming if empty string
         out MONameType name,
         in StringSetType packageNameList,
         in unsigned short priority)
             // conditional
             // priorityPackage
             // GET-REPLACE
         raises (itut_x780::ApplicationError,
              itut_x780::CreateError);

    }; // interface ProtectionUnitFactory



/**
```

4.6 Interfaces – Façade

Le comportement des interfaces façade est identique à celui des interfaces à granularité fine correspondantes. Pour cette raison, ces interfaces ne comportent pas de commentaires. Les lecteurs se réfèreront aux interfaces à granularité fine du 4.5 en ce qui concerne le comportement de ces interfaces façade.

Le présent paragraphe peut être omis dans le langage IDL si le système de gestion ne prend en charge que des interfaces à granularité fine.

```
*/

/**
```

4.6.1 ProtectionGroup_F

```
*/

    interface ProtectionGroup_F: itut_x780::ManagedObject_F
    {
         const boolean revertiveDefault = TRUE;

         OperationalStateType operationalStateGet
            (in MONameType name)
            raises (itut_x780::ApplicationError);
```

```
AvailabilityStatusSetType availabilityStatusGet
    (in MONameType name)
    raises (itut_x780::ApplicationError);


MONameSetType supportedByObjectListGet
    (in MONameType name)
    raises (itut_x780::ApplicationError);

void supportedByObjectListSet
    (in MONameType name,
    in MONameSetType objectList)
    raises (itut_x780::ApplicationError);

void supportedByObjectListAdd
    (in MONameType name,
    in MONameSetType objectList)
    raises (itut_x780::ApplicationError);

void supportedByObjectListRemove
    (in MONameType name,
    in MONameSetType objectList)
    raises (itut_x780::ApplicationError);


ProtectionGroupType groupTypeGet
    (in MONameType name)
    raises (itut_x780::ApplicationError);


boolean revertiveGet
    (in MONameType name)
    raises (itut_x780::ApplicationError);

void revertiveSet
    (in MONameType name,
    in boolean revertive)
    raises (itut_x780::ApplicationError);


unsigned short waitToRestoreTimeGet
    (in MONameType name)
    raises (itut_x780::ApplicationError);

void waitToRestoreTimeSet
    (in MONameType name,
    in unsigned short waitTime)
    raises (itut_x780::ApplicationError);


LockedInConditionType lockedInConditionGet
    (in MONameType name)
    raises (itut_x780::ApplicationError);

void lockedInConditionSet
    (in MONameType name,
    in LockedInConditionType condition)
    raises (itut_x780::ApplicationError);


CurrentProblemSetType currentProblemListGet
    (in MONameType name)
    raises (itut_x780::ApplicationError,
              NOprotectionAlarmPackage);
```

```
        void invokeProtection
            (in MONameType name,
            in PSwitchType switchType,
            in ProtectionUnitNameSeqType protectedUnitList,
            in ProtectionUnitNameSeqType protectingUnitList,
            out PSwitchActionResultType actionResult)
            raises (itut_x780::ApplicationError);

        void releaseProtection
            (in MONameType name,
            in PSwitchType switchType,
            in ProtectionUnitNameSeqType protectedUnitList,
            in ProtectionUnitNameSeqType protectingUnitList,
            out PSwitchActionResultType actionResult)
            raises (itut_x780::ApplicationError);


        MANDATORY_NOTIFICATION(
            Notifications, protectionSwitchReporting)
        MANDATORY_NOTIFICATION(
            itut_x780::Notifications, stateChange)

        CONDITIONAL_NOTIFICATION(
            Notifications, protectionAlarm,
            protectionAlarmPackage)
        CONDITIONAL_NOTIFICATION(
            itut_x780::Notifications, objectCreation,
            createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
            itut_x780::Notifications, objectDeletion,
            createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
            itut_x780::Notifications, attributeValueChange,
            attributeValueChangeNotificationPackage)

        };  // interface ProtectionGroup_F



/**
```

4.6.2 ProtectionUnit_F

```
*/

    interface ProtectionUnit_F : itut_x780::ManagedObject_F
    {
        boolean protectingGet
            (in MONameType name)
            raises (itut_x780::ApplicationError);

        MONameSetType reliableResourceListGet
            (in MONameType name)
            raises (itut_x780::ApplicationError);

        MONameSetType unreliableResourceListGet
            (in MONameType name)
            raises (itut_x780::ApplicationError);

        ProtectionStatusType protectionStatusGet
            (in MONameType name)
            raises (itut_x780::ApplicationError);
```

```
          unsigned short priorityGet
              (in MONameType name)
              raises (itut_x780::ApplicationError,
              NOpriorityPackage);

          void prioritySet
              (in MONameType name,
              in unsigned short priority)
              raises (itut_x780::ApplicationError,
              NOpriorityPackage);

     }; // interface ProtectionUnit_F


/**
```

4.7 Notifications

```
*/
/**
Cette interface contient les définitions des notifications émises par des objets
gérés.

Les utilisateurs de notifications qui souhaitent des notifications typées
doivent annexer les opérations ci-dessous à l'interface de notification M.3120,
s'il y en a une.

Les éditeurs de notifications et les souscripteurs désirant utiliser des
notifications structurées basées sur les opérations définies ci-dessous
devraient suivre les règles de construction et de lecture de la structure des
notifications définies dans la Rec. UIT-T Q.816.

*/
          interface Notifications
     {

/**
La notification protectionSwitchReporting est émise depuis l'objet
ProtectionGroup pour signaler tout événement de commutation de protection.

*/
       void protectionSwitchReporting
              (in ExternalTimeType              eventTime,
              in MONameType                     source,
                    // protection group
              in ObjectClassType                sourceClass,
                    // always ProtectionGroup
              in NotifIDType                    notificationIdentifier,
              in CorrelatedNotificationSetType  correlatedNotifications,
              in AdditionalTextType             additionalText,
              in AdditionalInformationSetType   additionalInfo,
              in MONameType                     reportedProtectionUnit,
              in ProtectionStatusSetType        oldProtectionStatus,
              in ProtectionStatusSetType        newProtectionStatus,
              in ProtectionDirectionType        protectionDirection);


/**
La notification protectionAlarm est émise depuis l'objet ProtectionGroup pour
signaler toute défaillance de mécanisme de protection ou de ressource de
protection.

*/
```

```
      void protectionAlarm
            (in ExternalTimeType              eventTime,
            in MONameType                     source,
               // protection group
            in ObjectClassType                sourceClass,
               // always ProtectionGroup
            in NotifIDType                    notificationIdentifier,
            in CorrelatedNotificationSetType  correlatedNotifications,
            in AdditionalTextType             additionalText,
            in AdditionalInformationSetType   additionalInfo,
            in ProbableCauseType              probableCause);

    }; // interface Notfications

/**
```

4.8 Corrélation de noms

```
*/

/**

Le module ci-après contient les informations de corrélation de noms.
*/
    module NameBinding
    {
/**
```

4.8.1 ProtectionGroup

```
*/
      module ProtectionGroup_ManagedElement
      {
            const string   superiorClass =
                  "itut_m3120::ManagedElement";
            const boolean  superiorSubclassesAllowed = TRUE;
            const string   subordinateClass =
                  "itut_m3120::ProtectionGroup";
            const boolean  subordinateSubclassesAllowed = TRUE;
            const boolean  managerCreatesAllowed = TRUE;
            const DeletePolicyType deletePolicy =
                  itut_x780::deleteOnlyIfNoContainedObjects;
            const string   kind = "ProtectionGroup";
      }; // module ProtectionGroup_ManagedElement

/**
```

4.8.2 ProtectionUnit

```
*/
      module ProtectionUnit_ProtectionGroup
      {
            const string   superiorClass =
                  "itut_m3120::ProtectionGroup";
             const boolean superiorSubclassesAllowed = TRUE;
            const string   subordinateClass =
                  "itut_m3120::ProtectionUnit";
            const boolean  subordinateSubclassesAllowed = TRUE;
            const boolean  managerCreatesAllowed = TRUE;
            const DeletePolicyType deletePolicy =
                  itut_x780::deleteOnlyIfNoContainedObjects;
            const string   kind = "ProtectionUnit";
      }; // module ProtectionUnit_ProtectionGroup


    }; // module NameBinding
```

```
/**
```

4.9 ProbableCauseConst

```
Ce module contient les valeurs constantes définies pour l'UID ProbableCause.
Les valeurs ci-dessous doivent compléter le module ProbableCauseConst M.3120.

*/

     module ProbableCauseConst
     {

/**
Les valeurs 81-100 sont réservées pour les causes probables liées aux alarmes
équipement.
*/
       const short protectionMechanismFailure = 81;
       const short protectingResourceFailure = 82;

     }; // module ProbableCauseConst



}; // module itut_m3120



#endif // _itut_m3120_amd1_idl_
```

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 et protocole Internet
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Genève, 2002